N93-19401

1992 NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM

JOHN F. KENNEDY SPACE CENTER
UNIVERSITY OF CENTRAL FLORIDA

AUTOMATICALLY CALIBRATING ADMITTANCES IN KATE'S AUTONOMOUS LAUNCH OPERATIONS MODEL

PREPARED BY: Dr. Steve Morgan

ACADEMIC RANK: Associate Professor

UNIVERSITY AND DEPARTMENT: Baylor University
Department of Engineering
and Computer Science

NASA/KSC

DIVISION: Engineering Development

BRANCH: Artificial Intelligence

NASA COLLEAGUE: Carrie Parrish
Peter Engrand

DATE: August 7, 1992

CONTRACT NUMBER: University of Central Florida
NASA-NGT-60002 Supplement: 8

Acknowledgements

Thank you, Bob Merchant, for all the technical guidance this summer. Your knowledge of today's and tomorrow's KATE models made the automatic admittance calibrator the practical tool that it is today. Thank you, collaborators Carrie Parrish and Peter Engrand, for helping chart my course and clear the way this summer administratively. Thank you especially, Carol Valdez, Kari Stiles, and Loren Anderson, for operating the NASA/ASEE Summer Faculty Fellowship Program so flawlessly. I can't remember when I've enjoyed working more.

Abstract

This report documents a 1000-line Symbolics LISP program that automatically calibrates all 15 fluid admittances in KATE's Autonomous Launch Operations (ALO) model. (KATE is Kennedy Space Center's Knowledge-based Autonomous Test Engineer, a diagnosis and repair expert system created for use on the space shuttle's various fluid flow systems.) As a new KATE application, the calibrator described here breaks new ground for KSC's Artificial Intelligence Lab by allowing KATE to both *control and measure* the hardware she supervises. By automating a formerly manual process, the calibrator: 1) saves the ALO model builder untold amounts of labor, 2) enables quick repairs after workmen accidently adjust ALO's hand valves, and 3) frees the modeler to pursue new KATE applications that previously were too complicated. Also reported are suggestions for enhancing the program: 1) to calibrate ALO's TV cameras, pumps, and sensor tolerances, and 2) to calibrate devices in other KATE models, such as the shuttle's LOX and Environment Control System (ECS).

Summary

Last summer, this author gather several of KATE's model builders' tools into a Model Verification Toolkit. Coding the design of one of those tools, an automatic admittance calibrator for KATE's Autonomous Launch Operations (ALO) model, became the topic of this summer's project. A two-week survey of ALO's knowledge bases revealed the 15 admittances to be calibrated. During the next two weeks, the calibrator's designer manually adjusted valves and pump speeds of the ALO model, recorded its pressures with a pencil, and solved admittance equations off-line, until he discovered the best procedure for measuring every admittance. The ensuing four-week coding and testing effort produced the automated admittance calibrator program attached as Appendix A. Its typical output appears as Appendix B. This report is its documentation. The KATE modeler who first proposed this summer project claims he is pleased to have it.

Table Of Contents

LIST OF ILLUSTRATIONS

LIST OF TABLES

I
BACKGROUND

1.1 KATE

Kennedy Space Center's Knowledge-based Autonomous Test Engineer (KATE) is an artificially-intelligent diagnosis and repair expert system. KATE oversees the operation of fluid flow systems like that pictured in Figure 1-1. Guided by knowledge of admittances in the system hardware, KATE predicts flows from actual pressure measurements, and vice versa. When her predictions differ significantly from actual measurements, she uses something like fault tree analysis to isolate the error to a single failed device. KATE's Autonomous Launch Operations (ALO) model even swaps out the failed component for a redundant one to keep a simulated launch on schedule. Others of KATE's models include LOX, a very detailed model of the shuttle's liquid oxygen tanking operations, and the Environmental Control System (ECS), a model of the shuttle crew's breathing air processing.

1.2 KATE's ALO-H2O Model

A photograph of KATE's ALO-H2O hardware model appears in Figure 1-1. A hardware scale model of the shuttle's oxygen tanking operations, it substitutes cryogenic liquid oxygen with water and pressurized nitrogen gas for greater safety. A computer-aided design rendering of this hardware (see Figure 1-2) serves as KATE's ALO knowledge base. It shows connections of all significant devices along with their current operating conditions. Unfortunately, the fluid flow components' 15 admittances are hidden from view inside the model's software, since these do not correspond one-to-one to the 79 individual illustrated devices. Meaningless details such as pipe dimensions and tees are omitted entirely from the model for simplicity.

1.3 Building KATE's Models

KATE's model builders are busy people. Creating a KATE model like ALO is first an exercise in computer-aided design (CAD). But CAD models seldom include the functional detail that KATE requires for error diagnosis. Furthermore, KATE is written in LISP, and CAD information must be painstakingly reformatted as knowledge-base frames. These frames amount to upwards of 15,000 lines of LISP code even for the simple ALO model shown in Figure 1-2. Obscure interactions among knowledge base elements consume considerable debugging time during the never-ending development of ever more complicated expert systems.

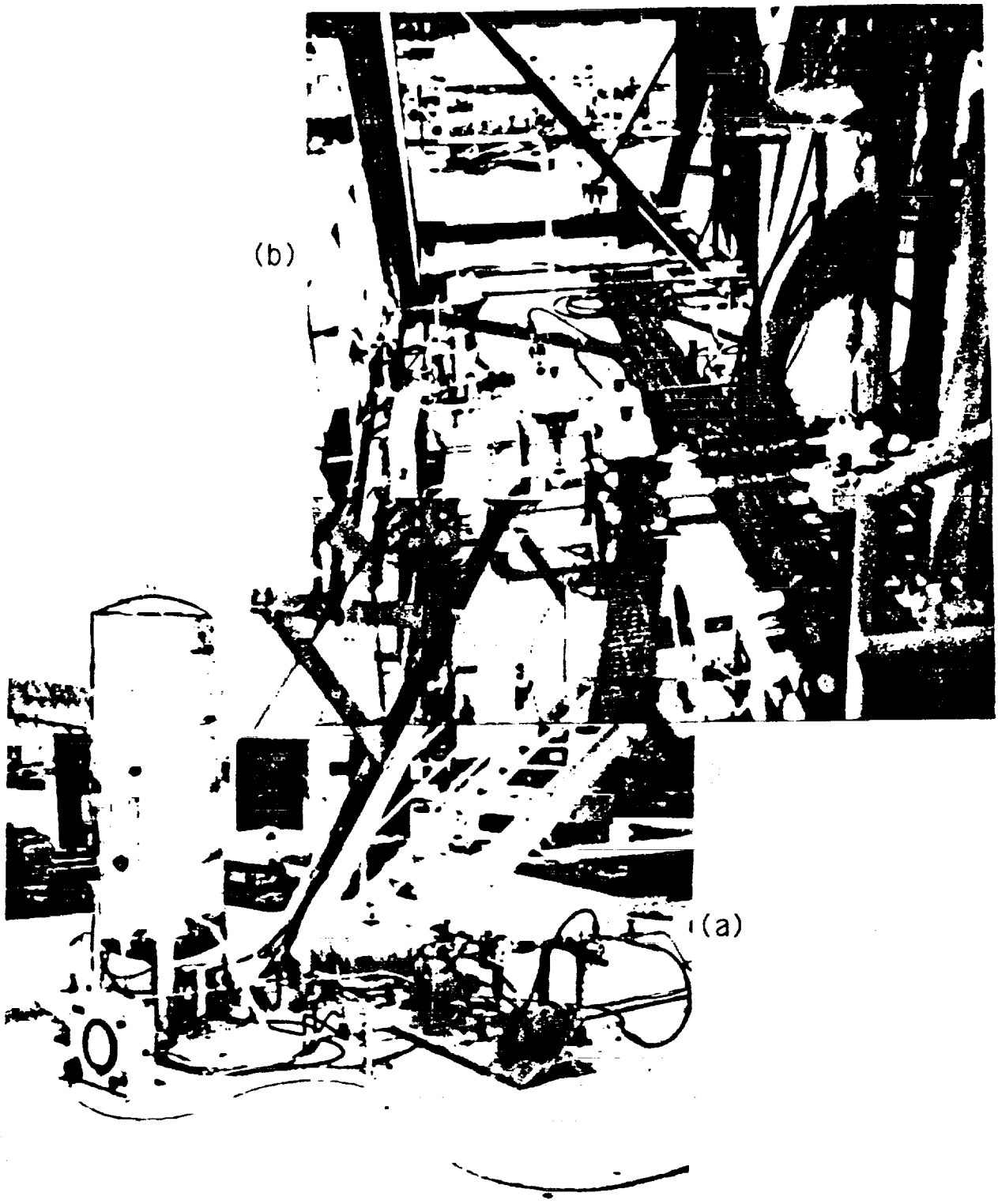

Figure 1-1. KATE's ALO-H2O Model Hardware: a) ground level pump circuit, b) vehicle tank circuit 20-feet up.

ORIGINAL PAGE IS
OF POOR QUALITY

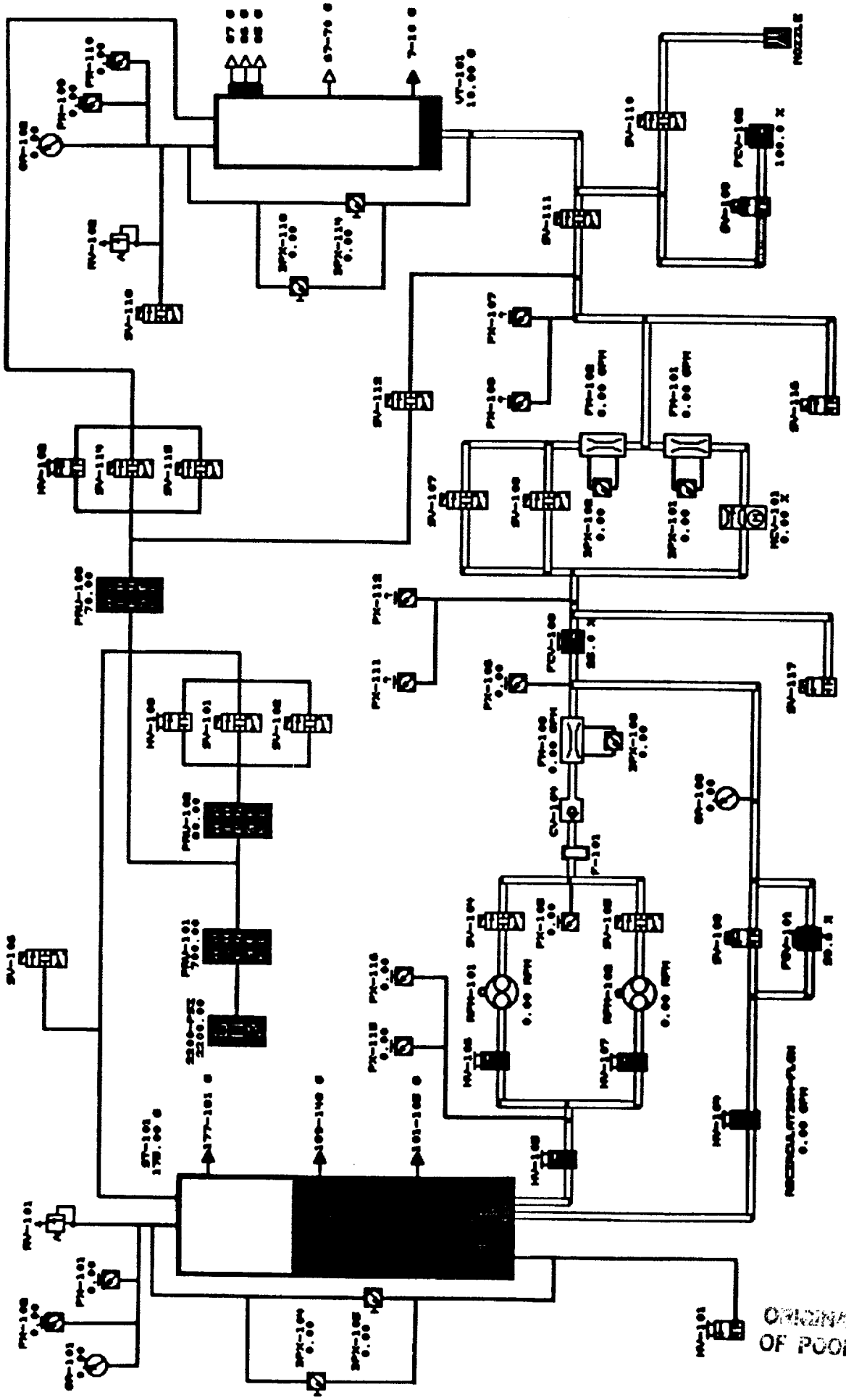

Figure 1-2. ALO User's Terminal Overview Screen

ORIGINAL PAGE IS
OF POOR QUALITY

1.4 KATE's Model Verification Tools

KATE's modelers frequently create little blocks of LISP code just to assist them in their efforts. In the summer of 1991, this author gathered several of these into a Model Verification Toolkit [1]. A linear regression algorithm, which fits straight lines to random model measurements in a least-squares sense, was one of the first model verification tools. Recently Boeing personnel have added a Knowledge Base Editor, which makes encoding CAD device data into KATE's frames much easier. Additional tools were proposed, including a program to automatically calibrate the ALO model's 15 admittances. That program became this summer's project. The toolkit is already saving modelers precious time and enabling them to apply KATE to more complicated and more powerful systems.

II
METHODS

2.1 KATE's Fluid-Flow Admittance Equation

KATE's admittance equation evolves from the law of energy conservation, which equates the potential energy of a pressure head across a pipe to the kinetic energy lost by fluid flowing against pipe friction. (See Figure 2-1 below.)

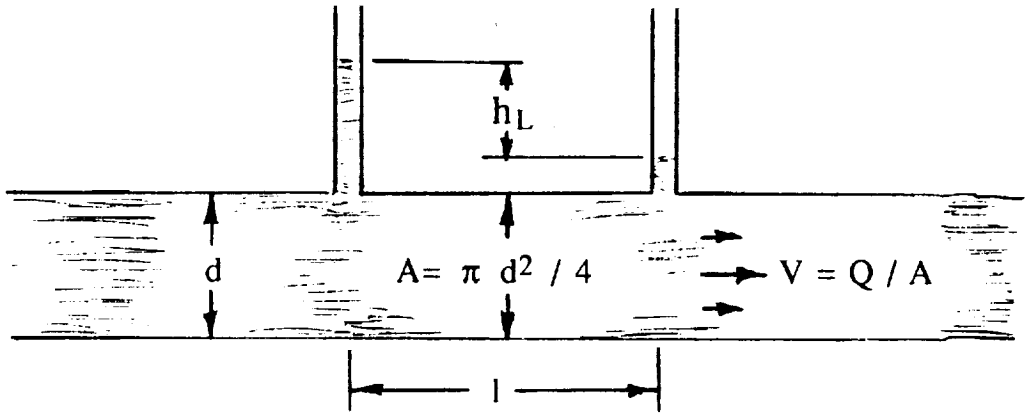

Figure 2-1. Energy Conserved By Fluid Flowing In A Pipe [2]

Darcy's equation [2] restates this basic law in terms of pipe dimensions:

$$h_L = f\, l\, V^2 / 2\, d\, g,$$

where h_L is pressure head in feet,
f is the dimensionless pipe friction factor,
l and d are pipe length and diameter in feet,
V is mean fluid velocity in feet/second,
and the gravitation constant, g, is 32.2 feet/sec^2.

Knowing that the specific weight of water is 62.37 pounds/foot^3 at 60°F, we can convert the head, h_L, to a pressure drop, ΔP, in psig:

$$h_L = \Delta P\ (144\ in^2/ft^2)\ /\ (62.37\ pounds/foot^3) = 2.309\ \Delta P.$$

Recognizing that fluid flow rate, Q, is the product of mean velocity, V, and pipe area, $\pi\ d^2\ /\ 4$, we can rewrite Darcy's formula as the ratio of flow rate and the square root of pressure drop:

$$V\ /\ \sqrt{h_L} = \sqrt{2\ g\ d\ /\ f\ l} = Q\ (4\ /\ \pi\ d^2)\ /\ \sqrt{2.309\ \Delta P}.$$

KATE lumps all of these pipe constants into one and calls them "admittance,"

$$A_i = (\pi\ d^2\ /\ 4)\ \sqrt{2\ g\ d\ 2.309\ /\ f\ l}$$
$$= Q\ /\ \sqrt{\Delta P}. \qquad (2.1)$$

The dimensions of admittance are ft^3-inches/minute-$\sqrt{pound}$. Instead of obtaining the admittance of a complicated pipe circuit from many length and diameter measurements, KATE finds it more convenient to measure flow rate and pressure drop and to solve equation 2.1 above.

2.2 Statistical Averaging

Actually, pressures and flows are measured at many pump speeds, and the several admittances are statistically averaged to reduce measurement noise. The average value (i.e., mean) of a sampled population is their sum divided by the number of samples taken. A small standard deviation (i.e., the root-mean-square of the samples' distances from their mean) may be regarded as a figure of merit on the random sampling process. When sampling from a normal, Gaussian population, the user can be assured that 95% of his samples will fall within ± 1 standard deviation (s.d.) of the mean. An excessive admittance s.d. (say 20% larger than the mean or more) warns of unreliable pressure or flow measurements. In such cases, the calibration should be done over.

2.3 ALO's "Readily Apparent" Admittances

Bob Merchant's April 1, 1992 Memo [3] details the proposal for ALO's automatic admittance calibrator. Only those "readily apparent" admittances carrying measurable flows and bracketed by measurable pressures are to be calibrated. Other admittances may be neglected, since they are of no use to KATE in predicting pressures and flows during failure diagnoses. The 15 admittances that meet this "readily apparent" definition are illustrated in an electronic analog of ALO in Figure 2-2. Those who are unfamiliar with the ALO model can easily locate traditional admittance values in ALO's knowledge base by refering to the right-hand two columns of Table 2-1.

2.4 Tank Pressurization

When the ALO model pressurizes (or depressurizes) the storage and vehicle water tanks, gas flow rate is not measured. But flow can be inferred from measured pressures of the tanks and their gas supplies (or vents) [3]. Tank pressures rise exponentially toward the supply pressure (or fall toward atmospheric pressure) asymptotically with time. According to the Gas Law, the amount of gas in a tank varies in proportion to its pressure. Thus, the time dependent amount of gas in either of ALO's tanks (in cubic feet) is:

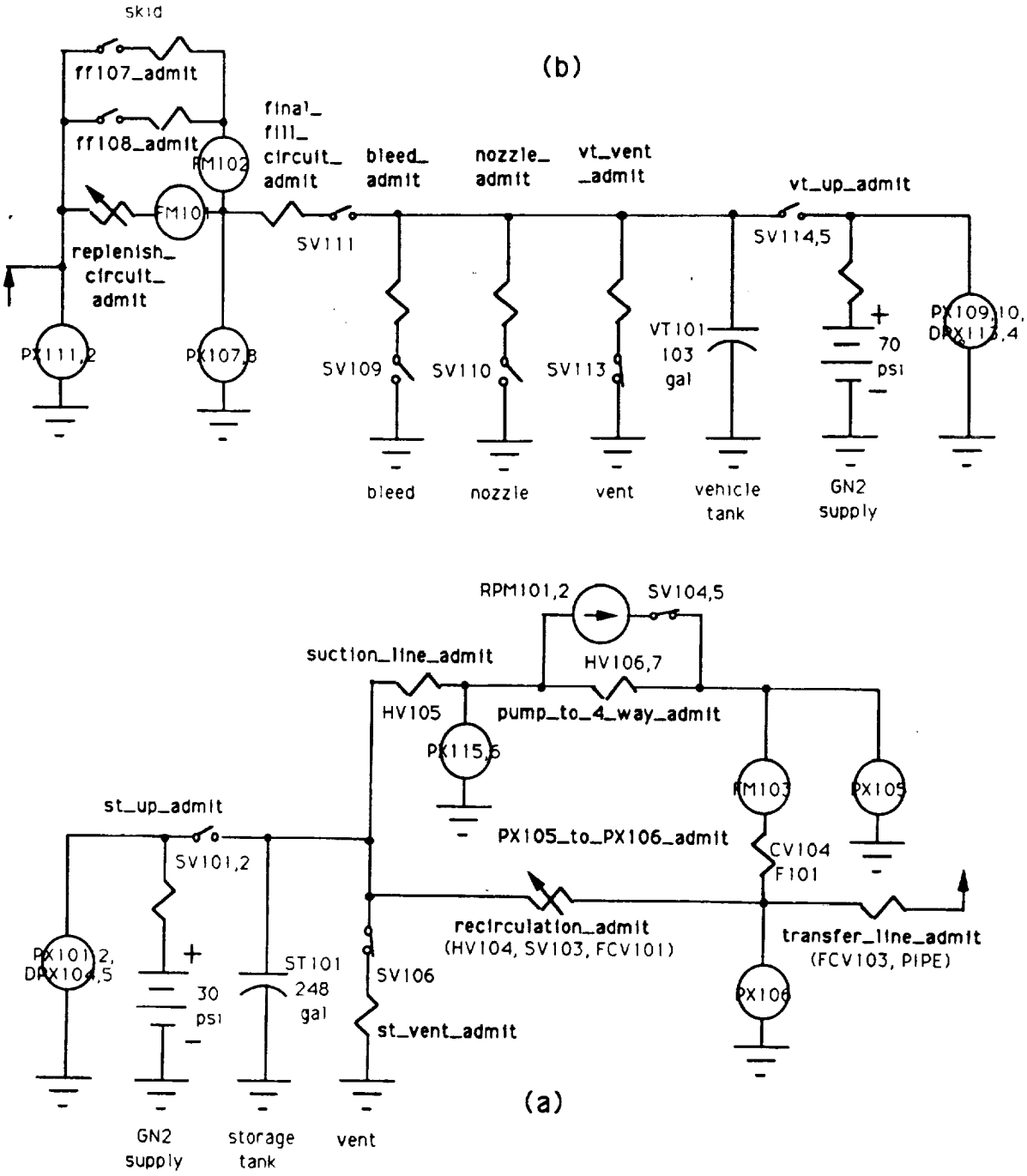

Figure 2-2. ALO's "Readily Apparent" Fluid Flow Admittances: a) ground level pump circuit, b) vehicle tank circuit 20-feet up.

Table 2-1
ALO's "Readily Apparent" Admittances

Admittance	Bracketing Pressures	Flow Meas.	Reference (page-line)	Trad'l Value
1. Tank Pressurization (Nitrogen Gas) Circuits....				
st-up-admit	30psig supply to storage tank	Fcn	8-16[4]	23.
vt-up-admit	70psig supply to vehicle tank	of	8-6[4]	18.5
st-vent-admit	Storage tank to atm via SV106	tank	9-14[4]	25.5
vt-vent-admit	Vehicle tank to atm via SV113	pres.	9-6[4]	53.
2. Pump Circuits....				
suction-line-admit	From storage tank to PX115	FM103	2-67[5]	15.5
pump-to-4-way-admit	From PX115 to PX105	FM103	1-39[6]	10.58
px105-to-px106-admit	From PX105 to PX106	FM103	new	4.65
pump-circuit-admit	Three admits above in series	-	2-38[5]	4.1*
recirculation-admit	PX106 to storage tank, SV103 closed	FM103	3-71[5]	1.64
	Ditto, but with SV103 open	FM103	3-71[5]	7
3. Vehicle Tank Circuits....				
ff107-admit	PX111 to PX107, only SV107 open	FM102	6-38[5]	6.47
ff108-admit	PX111 to PX107, only SV108 open	FM102	6-40[5]	6.47
fast-fill-circuit-admit	Two admits above in parallel	-	6-38[5]	6.47*
replenish-circuit-admit	MCV101 xfer function (e.g., 33%)	FM103	8-61[5]	0.02786x -0.7013
skid-admit	Two admits above in parallel	-	9-24[5]	6.69*
final-fill-circuit-admit	From PX107 to vehicle tank	FM102	1-16[5]	7.3
upper-fill-circuit-admit	Two admits above in series	-	12-26[5]	4.93*
transfer-line-admit	From PX106 (up 20 ft) to PX111	FM102	5-25[5]	2.36
tank-fill-admit	Two admits above in series	-	13-27[5]	2.13*
4. Vehicle Tank Drain Circuits....				
nozzle-admit	Vehicle tank to atm via SV110	FM102	14-21[5]	0.55
bleed-admit	Vehicle tank to atm via SV109	-flow$_{vt}$	14-9[5]	0.34

* Derived admittances are *functions* of measured ones (not constants) and they vary with valve positions.

$$N_i = P_i N_0 / P_0 , \qquad (2.2)$$

in which N_0 is the capacity of the empty tank,
P_0 is absolute atmospheric pressure, and
P_i is the measured tank pressure time function.

The ratio N_0 / P_0 must be measured when the tanks are empty, the only time ALO can accurately assess their ullage air-space volume.

2.5 Infering Gas Flow Rate

Given the amount of gas in the tank as a function of time, it is a simple matter to compute gas flow rate as a function of time:

$$Q_i = (N_i - N_{i-1}) / T_S \qquad (2.3)$$

in which Ts is the sampling time interval in minutes. KATE's 4-second (i.e., 1/15 minute) "heart beat" interval is a convenient sampling time interval. The pressure drop across the tank pressurization (or depressurization) circuit is the difference in the tank pressure and the source (or vent) pressure, $P_i - P_S$. Therefore, the circuit admittance time function is

$$\begin{aligned} A_i &= Q_i / \sqrt{P_i - P_S} \\ &= (N_0/P_0)\ (P_i - P_{i-1}) / (T_S \sqrt{P_i - P_S}). \end{aligned} \qquad (2.4)$$

For a given tank ullage, the ratio N_0/P_0T_S is constant. That is,

$$N_0/P_0T_S = 32.75 \text{ cuft/psi-min}$$

for the 32.75 cuft ullage of ALO's empty storage tank, and

$$N_0/P_0T_S = 13.77 \text{ cuft/psi-min}$$

for the 13.77 cuft ullage of ALO's empty vehicle tank. Thus,

$A_i = 32.75\ (P_i - P_{i-1}) / \sqrt{P_i - 30\text{psi}}$, for pressurizing the storage tank,
$A_i = 32.75\ (P_i - P_{i-1}) / \sqrt{P_i}$, for depressurizing the storage tank,
$A_i = 13.77\ (P_i - P_{i-1}) / \sqrt{P_i - 70\text{psi}}$, for pressurizing the vehicle tank, and
$A_i = 13.77\ (P_i - P_{i-1}) / \sqrt{P_i}$, for depressurizing the vehicle tank.

As before, averaging the calculated admittances, A_i, reduces measurement noise.

2.6 Motor Control Valve Transfer Function

KATE can set ALO's skid replenish valve MCV-101 to any desired position from fully closed to 100% open. Its variable admittance is characterized by the transfer function:

$$\text{Admittance} = \%\text{open} * \text{slope} + \text{admit}_0,\ r = \_\_\_\_\ \%. \tag{2.5}$$

In this function, the slope, admittance intercept, and correlation coefficient, r, are defined by the least-squared-error linear regression equations [7]:

$$\text{slope} = \frac{n\,\Sigma xy - \Sigma x\,\Sigma y}{n\,\Sigma x^2 - (\Sigma x)^2}, \tag{2.6}$$

$$\text{admit}_0 = \frac{\Sigma y - \text{slope}\,\Sigma x}{n}, \text{ and} \tag{2.7}$$

$$\text{correlation, } r = \frac{n\,\Sigma xy - Sx\,Sy}{\sqrt{\{[n\,\Sigma x^2 - (\Sigma x)^2][n\,\Sigma y^2 - (\Sigma y)^2]\}}}, \tag{2.8}$$

in which x and y represent the %open variable (typical range: 0.30 to 0.40) and the valve's admittances at those %open valve settings respectively. The correlation coefficient, r, tells the modeler the linearity (a figure of merit) of the transfer function. Correlations of 99.9% are common for this valve. Correlations less than 95% should prompt the modeler to take a new set of measurements.

2.7 Coding The Admittance Calibrator

Pressures must be sampled from the ALO model before the equations above can yield the 15 desired admittances. These pressures arise from plausible flows induced in some fluid circuit by one of ALO's two pumps. Collecting pressure samples is a simple matter of calling the LISP function, GET-CURRENT-VALUE, every four seconds with a pressure transducer's CAD reference designator as the function argument [8]. For example,

```
(L (SETQ x (GET-CURRENT-VALUE 'PX-115)))
```

evaluates actual pump head pressure when called by a control procedure. The pumps respond to motor speed commands, such as:

```
(C RPM-101 (3600 rpm) "pump rapidly"),
```

and ALO's valves respond to control procedure calls like these:

(C SV-103 OPEN "quicken recirculation flow")

(C SV-111 CLOSED "prevent fluid loss during tank pressurization")

(C MCV-101 (0.40 %) "replenish vehicle tank boil off").

Table 2-2 shows the best valve and pump settings, as well as flows and pressures to be measured for every ALO admittance. This table and the equations above constitute the automated admittance calibrator's software design. Its LISP code [9] and data output [10] are attached as Appendices A and B. Its operating instructions are:

1. Select CONTROL PROCEDURES on KATE's overview menu.
2. Left-click on DO-IT-ALL.

Table 2-2
ALO's Admittance Measurement Environments

Admittance	Valves Open/Closed* SV-1xx											MCV	Pump	Flow	Pressure Meas.
	01	03	04	06	07	08	09	10	11	13	14	-101	Speed	FM10x	upstr-dnstr+Δelev
Tank Pressurization and Depressurization Procedures....															
st-up-admit	O	C	C	C	C	C	C	C	C	C	O	.0	0	N_i-N_{i-1}	30psig-PX101+0"
vt-up-admit	O	C	C	C	C	C	C	C	C	C	O	.0	0	divided	70psig-PX109+0"
st-vent-admit	C	C	C	O	C	C	C	C	C	O	C	.0	0	by	PX101-0psig+0"
vt-vent-admit	C	C	C	O	C	C	C	C	C	O	C	.0	0	Ts	PX109-0psig+0"
FastFlow Procedure....															
suction-line-admit	C	C	O	O	O	C	C	C	O	O	C	.0	2500-3500	3	PX101+DPX104-PX115-5"
pump-to-4-way-admit	C	C	O	O	O	C	C	C	O	O	C	.0	2500-3500	3	PX105-PX115+19"
PX105-to-PX106-admit	C	C	O	O	O	C	C	C	O	O	C	.0	2500-3500	3	PX105-PX106+24"
transfer-line-admit	C	C	O	O	O	C	C	C	O	O	C	.0	2500-3500	2	PX106-PX111-236"
ff-107-admit	C	C	O	O	O	C	C	C	O	O	C	.0	2500-3500	2	PX111-PX107-1"
final-fill-circuit-admit	C	C	O	O	O	C	C	C	O	O	C	.0	2500-3500	2	PX107-DPX113-PX109-3"
FF108 and MCV101 Procedures....															
ff-108-admit	C	C	O	O	C	O	C	C	O	O	C	.0	3500-2500	2	PX111-PX107-1"
replenish-circuit-admit	C	O	O	O	C	C	C	C	O	O	C	.3-.4	3500	1	PX111-PX107-1"
Nozzle and Bleed Procedures....															
nozzle-admit	C	C	O	O	O	C	C	O	O	O	C	.0	3500-2500	2-vtin	PX109+DPX113-0psig-0"
bleed-admit	C	C	O	O	O	C	O	C	O	O	C	.0	2500-3500	2-vtin	PX109+DPX113-0psig-0"
Recirc Procedure....															
recirculation-admit	C	O/C	O	O	C	C	C	C	C	O	C	.0	3500-2500	3	PX106-PX101-DPX104

* KATE initially *closes* all solenoid and motor control valves.

III
RESULTS

3.1 ALO's Traditional (Standard) Vs. Automatically Calibrated Admittances

The first and last pages of the automatic admittance calibrator's 25-page output file [10] appear here as Appendix B. Table 3-1 compares these 15 automatically calibrated admittances with the traditional ones, taken as standard values. The right-hand column (see "Meas. Error") is the automatically calibrated admittance minus the traditional one, all divided by the traditional one. Significant measurement errors are discussed below.

3.2 Tank Pressurization And Depressurization Admittances

ALO calculates the storage and vehicle tank ullage pressure and vent admittances afresh every time they are used, because their equations include tank ullage-space variables. (Section 2.4 suggests that a better policy might be calibrating these constant admittances once and for all with both tanks empty.) Table 3-1 shows traditional (standard) values of 23, 18.5, 25.5, and 53 for these four admittances. Their automatically calibrated counterparts are about 1/10th of these amounts. Close inspection of ALO's knowledge base equations [4] reveals assumed ullage spaces for both tanks that are about 10 times actual size. Though this knowledge base error has been in place since ALO's beginning, it probably has not affected ALO's performance much. Gas flow has little effect upon ALO's (more important) modeled fluid flow. Standard deviations (sd's) of all tank pressurization admittances are 29% of their means. These excessive sd's probably result from the fact that only three samples were taken in calibrating each admittance.

3.3 Pump Circuit Admittances

Some large errors among ALO's five pump circuit admittances demand explanation. Most striking is the -72% error in pump-to-4-way-admit. ALO's knowledge base [6] traditionally defines this admittance as extending from the pump inlet, where there is no pressure transducer, to transducer PX-105. Consistent with the "readily apparent" principle advanced in section 2.2 above, its automated counterpart extends from pressure transducer PX-115 to PX-105. Hand valve HV-106 is excluded from the standard admittance but included in the automated one. Explaining that ALO is "still under development," ALO's modeler claims that he prefers the automated value over the traditional standard. Soon ALO's knowledge base will change to reflect pump-to-4-way-admit's new "readily apparent" definition instead. The greater accuracy of the composite pump-circuit-admit suggests that the other three component admittances in the pump circuit also may contradict our new "readily apparent"

III
RESULTS

3.1 ALO's Traditional (Standard) Vs. Automatically Calibrated Admittances

The first and last pages of the automatic admittance calibrator's 25-page output file [10] appear here as Appendix B. Table 3-1 compares these 15 automatically calibrated admittances with the traditional ones, taken as standard values. The right-hand column (see "Meas. Error") is the automatically calibrated admittance minus the traditional one, all divided by the traditional one. Significant measurement errors are discussed below.

3.2 Tank Pressurization And Depressurization Admittances

ALO calculates the storage and vehicle tank ullage pressure and vent admittances afresh every time they are used, because their equations include tank ullage-space variables. (Section 2.4 suggests that a better policy might be calibrating these constant admittances once and for all with both tanks empty.) Table 3-1 shows traditional (standard) values of 23, 18.5, 25.5, and 53 for these four admittances. Their automatically calibrated counterparts are about 1/10th of these amounts. Close inspection of ALO's knowledge base equations [4] reveals assumed ullage spaces for both tanks that are about 10 times actual size. Though this knowledge base error has been in place since ALO's beginning, it probably has not affected ALO's performance much. Gas flow has little effect upon ALO's (more important) modeled fluid flow. Standard deviations (sd's) of all four tank pressurization admittances are 29% of their means. These excessive sd's probably result from the fact that only three samples were taken in calibrating each admittance.

3.3 Pump Circuit Admittances

Some large errors among ALO's five pump circuit admittances demand explanation. Most striking is the -72% error in pump-to-4-way-admit. ALO's knowledge base [6] traditionally defines this admittance as extending from the pump inlet, where there is no pressure transducer, to transducer PX-105. Consistent with the "readily apparent" principle advanced in section 2.2 above, its automated counterpart extends from pressure transducer PX-115 to PX-105. Hand valve HV-106 is excluded from the standard admittance but included in the automated one. Explaining that ALO is "still under development," ALO's modeler claims that he prefers the automated value over the traditional standard. Soon ALO's knowledge base will change to reflect pump-to-4-way-admit's new "readily apparent" definition instead. The greater accuracy of the composite pump-circuit-admit suggests that the other three component admittances in the pump circuit also may contradict our new "readily apparent"

Table 3-1
The Answers

Admittance	Traditional Admittance	Auto Cal Admit (mean +/- sd)	Meas. Error
1. Tank Pressurization (Nitrogen Gas) Circuits....			
st-up-admit	23	2.280±.6677	10.1x
vt-up-admit	18.5	1.842±.5394	10.0x
st-vent-admit	25.5	2.485±.7279	10.3x
vt-vent-admit	53	5.077±1.487	10.4x
2. Pump Circuits....			
suction-line-admit	15.5	14.62±6.179	-6%
pump-to-4-way-admit	10.58	2.922±1.235	-72%
px105-to-px106-admit	4.65	5.701±2.409	+23%
pump-circuit-admit	4.1*	2.560*	-38%
recirculation-admit, SV103 closed	1.64	1.696±.7166	+3%
ditto but with SV103 open	7	6.213±2.595	-11%
3. Vehicle Tank Circuits....			
ff107-admit	6.47	6.066±2.564	-6%
ff108-admit	6.47	6.353±2.685	-2%
fast-fill-circuit-admit	6.47*	6.066*	-6%
replenish-circuit-admit (a transfer function)	0.02786x -0.7013	0.02781x-0.7, r=1.	0% @33%open
skid-admit	6.69*	5.375*	-20%
final-fill-circuit-admit	7.3	6.977±2.949	-4%
upper-fill-circuit-admit	4.93*	4.258*	-14%
transfer-line-admit	2.36	2.360±.9975	0%
tank-fill-admit	2.13*	2.064*	-3%
4. Vehicle Tank Drain Circuits....			
nozzle-admit	0.55	0.02286±.02158	-96%
bleed-admit	0.34	0.06923±.02797	-80%

* Derived admittances are *functions* of measured ones (not constants), and they vary with valve positions.

principle. Sd's of the pump circuit admittances range from 42 to 44% of their means.

3.4 Vehicle Tank Circuits

Component admittances in the vehicle tank circuit show smaller errors. A dead-level perfect overall tank circuit admittance (i.e., tank-fill-admit) suggests minor changes in admittance definitions again, as ALO's modeler embraces the "readily apparent" concept. He dismisses these errors (all $\leq$ 20%) as "insignificant." One might argue that automated admittances reported here should agree *exactly* with traditional ones. All pressures and flows the calibrator measured were *modeled* values, arising from traditional admittances in ALO's knowledge base. Actually, ALO adds a "realistic" offset [11] to every modeled pressure before displaying it on the user's overview screen and before delivering it to the user's program via GET-CURRENT-VALUE. In the range -0.72 to +0.34 psi, these small offsets add a fixed measurement error that the admittance calibrator's statistical averaging cannot eliminate. A -0.72 psi offset on a 2.4 psi pressure measurement could account for a +20% error in admittance. Sd's of all four constant vehicle tank admittances are 42% of their means.

3.5 Nozzle And Bleed Admittances

Automatic calibration of nozzle and bleed valve admittances presents some interesting design problems. The space shuttle's engine nozzle valve never opens during oxygen tanking, and the bleed valve only opens occasionally, when the tank has been overfilled. One might logically wonder why ALO models these valve admittances at all, since it is preoccupied with device failures during tanking. Evidently, these admittances lie at the edge of ALO's knowledge domain -- bizarre errors lie waiting there for the unwary. For example, 10.13 gallons of water typically flows from the storage tank to the vehicle tank every minute during the fast-fill oxygen tanking procedure. When the nozzle valve opens, and an unmeasured portion (this path has no flowmeter) of that 10.13 gpm flows out the nozzle, the vehicle tank's inlet flow *rises* to 10.30 gpm! Final-fill-admit, nozzle-admit, and bleed-admit all have solenoid valve SV-111 in common, so combining two or more of these admittances is meaningless. As the vehicle tank approaches half-full with the bleed valve open, its inlet flow typically approaches zero and then reverses, augmenting the bleed path's pumped flow. The paradoxical negative admittances that arise at these times suggest confusion in the knowledge base. In the light of this discussion, maybe a 96% error is not so bad. Sd's of the nozzle and bleed admittances are 94 and 40% of their means.

3.6 Error Summary

Causes of calibrator errors range from ALO "realistically" offsetting modeled pressures to variant admittance definitions to errors in ALO's evolving

knowledge base equations. ALO's modeler claims to be better satisfied with the automatically calibrated admittances than with the traditional ones. And customer satisfaction is the ultimate test of any automated productivity aid. Statistics (sd's) of all admittances will improve when procedure compiler enhancements (currently underway) permit looping a dozen times on a variable pumping speed.

IV
CONCLUSIONS

4.1 Enhanced Modeler Productivity

Perhaps it is stretching a point to compare the automated admittance calibrator's 8-minute run time with those 6 to 8 months ALO's modeler spent filling his 63-page notebook [13] with manual admittance calculations. (He also developed other parts of the model in that same time interval.) But there is no denying that automated admittance calibration does save ALO's modeler considerable time and effort. He is pleased to have it. It deserves to be included among the other modeler productivity aids in KATE's Model Verification Toolkit. Creating it has been an illuminating experience. This author has a deeper understanding of the problems KATE's modelers encounter and how to help them.

4.2 Clarified ALO Model

In the process of opening their minds to students, teachers often discover ways of improving their own reasoning processes. ALO's traditional admittances came from painstaking manual measurements of pumps, valves, pipes, tees, and elbows (and complicated combinations of these) on the space shuttle launch pad. ALO's modeler came up with the "readily apparent" admittance definition specifically to guide this newcomer in developing the automatic admittance calibrator. These 15 easily measured admittances (i.e., flow rates divided by square roots of pressure drops), plus a scant few of their series and parallel combinations, could replace the dozens of arbitrarily defined admittances that now litter ALO's knowledge base. His fresh look at ALO has clarified the model builder's task.

4.3 KATE's Philosophical Advancement

In the automated admittance calibrator, we have an novel application of KATE in which she changes hardware systems and then measures the effects of those changes. Two years ago, ALO's modeler broke new ground by enabling KATE to control environments she formerly observed passively. Now we have gone a step further toward making KATE a master of hardware environments. As her ALO model knowledge base measures its own admittances, KATE shows considerably more introspection than the average computer program. Could we speculate that KATE has achieved artificially intelligent "self-awareness"?

V
RECOMMENDATIONS

5.1 ALO's Future

ALO's automatic admittance calibrator has proved itself in a *modeled* environment. (Simulated by ALO's software, the pressures and flows the calibrator measured did not come from real hardware.) Now it is ready to be attached to the real hardware model (see Figure 1-1) in KSC's Large Equipment Test Facility (LETF). If a repeated Chapter 3 error analysis on real hardware is successful, ALO's modeler should arrange for the calibrator to stuff its 15 admittances directly into the knowledge base. If automatically calibrating ALO's admittances improves both model quality and modeler productivity, could calibrating other items be beneficial? Yes, indeed. ALO's TV cameras, pumps, and sensor tolerances should be calibrated next, as described below. Later this portable tool can be installed in KATE's LOX and ECS models to reap more productivity gains.

5.2 Automatically Calibrating ALO's Cameras

Occasionally ALO's two television cameras get bumped, and then they don't aim at failed devices as they should. A user-interactive calibration procedure is needed to trim camera pan, tilt, and focus upon demand. The procedure is summarized in the software design of Table 5-1. Because it relies heavily upon hardware, this procedure should be coded and tested after ALO is reinstalled in KSC's LETF area.

Table 5-1
TV Camera Calibration Procedure

1. Notice which device icon the mouse is pointing at.
2. Slew the camera to the coordinates already given in the device frame's *cam-coord* slot.
3. Pause while the user bumps the camera servo joy stick to manually trim its direction and focus.
4. Automatically read the camera's actual position from its servo hardware.
5. Write these new camera coordinates into the device frame's slot.

5.3 Automatically Calibrating ALO's Pumps

KATE needs to know how much head pressure to expect from ALO's pumps for any given rpm command. Pumps age, so automating calibration of the pumps' rpm-to-pressure transfer function can be expected to improve both

model performance and modeler productivity by permitting more frequent, and less time consuming updates. Table 5-2 summarizes a procedure for calibrating ALO's pumps. A coded but untested control procedure for pump calibration appears in Appendix A (see ".pump-data"). This procedure illustrates some experimental code that the calibrator's other procedures do not have. To improve portability to other KATE models, a disk file carries raw pressure measurements from a data-taking control procedure to a separate LISP analysis function. To simplify procedure coding, an improved looping macro (see "REPEAT") allows a *variable* to control pump speed. Control procedure compiler enhancements that are underway now will make these calibrator improvements workable.

Table 5-2
Pump Calibration Procedure

1. Configure ALO's valves the same as "pump-to-4-way" in Table 2-2.
2. Run pump from 1800 to 3600 rpm in 100 rpm steps.
3. Measure pump head pressure (PX105 minus PX115) in every step.
4. Use linear regression to obtain a straight line transfer function that relates commanded rpm values to measured pump head pressure.
5. If the linear regression algorithm reports a correlation poorer than 95%, check the pump and do it again.
6. Stuff the transfer function into the pump's knowledge base frame.

5.4 Calibrating Sensor Tolerances

ALO's modeler needs a sensor tolerance calibrator to tell him when tolerances get too sloppy. Device failures trigger KATE's diagnoser when actual measurements differ from their modeled values by more than a prescribed tolerance. KATE's modelers typically respond to false alarms (i.e., diagnoses triggered by out-of-tolerance sensors, not device failures) by arbitrarily widening sensor tolerances. Tolerances have been known to grow so large that device failure does not trigger the diagnoser. (Such growth admittedly is rare, because the catastrophic device failures that KATE is capable of diagnosing affect measurements dramatically.) The sensor tolerance calibration procedure in Table 5-3 below was inspired by conversation with Steve Beltz [13].

Table 5-3
Sensor Tolerance Calibration Procedure

1. Collect a complete set of measured-modeled differences for all sensors every 4-seconds (i.e., every KATE "heart beat").
2. After collecting 100Kbytes or more of these, alarm the user every time a measured-modeled difference exceeds $\pm$99 percentile.
3. After collecting 1Mbytes of these, display their histograms.

4. If the user approves, stuff each of the 99-percentile limits into its sensor tolerance slot in the model knowledge base.
5. Stop collecting measured-modeled differences:
    a. Anytime the user wants to.
    b. For 7 seconds after any abrupt control procedure change.
    c. While a sensor is "nuked" (i.e., the knowledge base's way of telling the diagnoser to ignore a temporarily dubious measurement).
6. Display histograms and flush the accumulated measured-modeled differences prematurely, if the user demands.

APPENDIX A

AUTOMATED ADMITTANCE CALIBRATOR PROGRAM LISTING

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: KATE; Base: 10 -*-


(SETF *ADMIT-FILE* "G:>MORGAN>ANSWERS.LISP")

;;; KATE-ALO's Initial State:
;;;   Hand valves, HV-104, -105, -106, -107, CV-104 all open.
;;;   Flow-control hand valves, FCV-101 = 30%, FCV-103 = 25%, FCV-102 = 100%.
;;;   All solenoid and motor control valves closed.
;;;   ST-101 = 175gal, VT-101 = 10gal.

(defun format2 (arg)
  (zwei:with-editor-stream (admit-stream :buffer-name *ADMIT-FILE*)
    (format admit-stream "~A~%" arg))
  arg)

;(defun format2 (arg)   ;;;  replace PRINTs with format2 to redirect output to the ANSWERS file
;  (format *rprt* "~A~%" arg))

;(defun format2 (arg)
;  (print arg))

(defvar *.ST-ULLAGE* nil "volume of storage tank air space: 0.0 to 245.8/7.481 cu.ft.")
(defvar *.VT-ULLAGE* nil "volume of vehicle tank air space: 0.0 to 100.0/7.481 cu.ft.")
(defvar *.rpm*  nil "current pump speed command: 1800 - 3600 rpm")
(defvar *.opencmd*  nil "current control valve position command: 0.0 to 1.0")
(defvar *.valve*  nil "current control valve of interest: MCV-101 or FCV-101")
(defvar *.st-up-admit*  nil "storage tank ullage pressurizing admittance")
(defvar *.vt-up-admit*  nil "vehicle tank ullage pressurizing admittance")
(defvar *.st-vent-admit*  nil "storage tank ullage venting admittance")
(defvar *.vt-vent-admit*  nil "vehicle tank ullage venting admittance")
(defvar *.suction-line-admit*  nil "storage tank to pump circuit admittance")
(defvar *.pump-to-4-way-admit*  nil "pump to PX-105 admittance")
(defvar *.PX105-to-PX106-admit*  nil "PX-105 to PX-106 admittance")
(defvar *.transfer-line-admit*  nil "PX-106 to skid admittance")
(defvar *.ff-107-admit*  nil "fast fill valve SV-107 admittance")
(defvar *.ff-108-admit*  nil "fast fill valve SV-108 admittance")
(defvar *.final-fill-circuit-admit*  nil "PX-107 to vehicle tank admittance")
(defvar *.replenish-circuit-admit*  nil "skid's slow fill admittance")
(defvar *.nozzle-admit*  nil "vehicle tank to nozzle admittance")
(defvar *.bleed-admit*  nil "vehicle tank to bleed outlet admittance")
(defvar *.recirculation-admit*  nil "PX-106 to storage tank backflow admittance")
(defvar *.pump-circuit-admit* nil "series pump-to-4-way-admit,suction-line-admit,PX105-to-PX106-admit")
(defvar *.fast-fill-circuit-admit*  nil "ff-107-admit and ff-108-admit in parallel")
(defvar *.skid-admit*  nil "replenish-circuit-admit and fast-fill-circuit-admit in parallel")
(defvar *.upper-fill-circuit-admit*  nil "skid-admit and final-fill-circuit-admit in series")
(defvar *.tank-fill-admit*  nil "upper-fill-circuit-admit and transfer-line-admit in series")
(defvar *.FM-103* nil "list of 30 FM-103 samples taken in 2 minutes")
(defvar *.FM-103sloflo* nil "list of 3 FM-103 samples taken with SV-103 closed")
(defvar *.FM-103fasflo* nil "list of 3 FM-103 samples taken with SV-103 open")
(defvar *.FM-102* nil "list of 30 FM-102 samples taken in 2 minutes")
(defvar *.FM-101* nil "list of 30 FM-101 samples taken in 2 minutes")
(defvar *.%open* nil "list of 10 control valve %open samples taken in 2 minutes")
(defvar *.PX-101* nil "list of 30 PX-101 samples taken in 2 minutes")
(defvar *.PX-109* nil "list of 30 PX-109 samples taken in 2 minutes")
(defvar *.DPX-104* nil "list of 30 DPX-104 samples taken in 2 minutes")
(defvar *.PX-115* nil "list of 30 PX-115 samples taken in 2 minutes")
(defvar *.PX-105* nil "list of 30 PX-105 samples taken in 2 minutes")
(defvar *.PX-106* nil "list of 30 PX-106 samples taken in 2 minutes")
(defvar *.PX-106sloflo* nil "list of 3 PX-106 samples taken with SV-103 closed")
(defvar *.PX-106fasflo* nil "list of 3 PX-106 samples taken with SV-103 open")
(defvar *.PX-111* nil "list of 30 PX-111 samples taken in 2 minutes")
(defvar *.PX-107* nil "list of 30 PX-107 samples taken in 2 minutes")
(defvar *.DPX-113* nil "list of 30 DPX-113 samples taken in 2 minutes")
(defvar *.a1* nil "temp space in FastFlow")
(defvar *.a2* nil "temp space in FastFlow")
(defvar *.a3* nil "temp space in FastFlow")
(defvar *.b1* nil "temp space in FastFlow")
(defvar *.b2* nil "temp space in FastFlow")
(defvar *.b3* nil "temp space in FastFlow")
(defvar *.c1* nil "temp space in FastFlow")
(defvar *.c2* nil "temp space in FastFlow")
(defvar *.c3* nil "temp space in FastFlow")
(defvar *.d1* nil "temp space in FastFlow")
(defvar *.d2* nil "temp space in FastFlow")
(defvar *.d3* nil "temp space in FastFlow")
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
(defvar *.e1* nil "temp space in FastFlow")
(defvar *.e2* nil "temp space in FastFlow")
(defvar *.e3* nil "temp space in FastFlow")
(defvar *.f1* nil "temp space in FastFlow")
(defvar *.f2* nil "temp space in FastFlow")
(defvar *.f3* nil "temp space in FastFlow")
(defvar *.g1* nil "temp space in FastFlow")
(defvar *.g2* nil "temp space in FastFlow")
(defvar *.g3* nil "temp space in FastFlow")
(defvar *.h1* nil "temp space in FastFlow")
(defvar *.h2* nil "temp space in FastFlow")
(defvar *.h3* nil "temp space in FastFlow")
(defvar *.i1* nil "temp space in FastFlow")
(defvar *.i2* nil "temp space in FastFlow")
(defvar *.i3* nil "temp space in FastFlow")
(defvar *.j1* nil "temp space in FastFlow")
(defvar *.j2* nil "temp space in FastFlow")
(defvar *.j3* nil "temp space in FastFlow")
(defvar *.k1* nil "temp space in FastFlow")
(defvar *.k2* nil "temp space in FastFlow")
(defvar *.k3* nil "temp space in FastFlow")
(defvar *.slow-recirc* nil "recirculation line admittance with SV-103 closed")
(defvar *.fast-recirc* nil "recirculation line admittance with SV-103 open")
(defvar *.pump-speed* nil "current pump speed during calibration")
(defvar *.pmp-spd-list* nil "all 19 pump speeds during calibration")
(defvar *.pump-xfer-fcn* nil "calibrated transfer function of RPM-101")
(defvar *.FM103admit* 3.75  "admittance of FM103 flow meter venturi")
(defvar *.FM102admit* 3.63  "admittance of FM102 flow meter venturi")
(defvar *.FM101admit* 0.667 "admittance of FM101 flow meter venturi")
(defvar *.in-to-psi*  27.7   "divide by this to convert inches of H2O to psi")
(defvar *.gal-to-cuft* 7.481 "divide by this to convert gallons to cubic feet")
(defvar *.ST-cap* 245.8 "storage tank capacity in gallons")
(defvar *.VT-cap* 103.0 "vehicle tank capacity in gallons")
(defvar *.ST-supl* 30 "storage tank's GN2 supply pressure")
(defvar *.VT-supl* 70 "vehicle tank's GN2 supply pressure")
(defvar *.atm* 0 "atmospheric pressure in psig")

;;;  Look for these on KATE's mouseable Control Procedures menu....
(setf *SYSTEM-PROCEDURES* '(.pressurize .depressurize .fastflow .ff108 .mcv101 .nozzle .bleed .recirc .de
rived .done .do-it-all bob))

;;;   Bob's ALO-KB Caller....
(defun GET-TANK-ULLAGE (tank-capacity amt)
  "Input is tank-capacity (in gal) and amt of fluid in tank (in gal),
   output is ullage space in ft3"
  (/ (- tank-capacity amt) 7.481))

;;;   ************************************************************************************************

;;;   **********************   Scott's Regression Line Drawing Package      ****************************

;;;   ************************************************************************************************

(defflavor graph-pane (mxx mnx mxy mny scale)
           (tv::window)
  :settable-instance-variables
  :readable-instance-variables)

(defvar *graph-pane* nil)

(setq *graph-pane*
      (tv::make-window 'graph-pane
                       :size '(500 500)
                       :blinker-p nil
                       :save-bits t
                       :label nil))

(defmethod (scale graph-pane)(xl yl)
  (setq mxx (apply #'max xl)
        mnx (apply #'min xl)
        mxy (apply #'max yl)
        mny (apply #'min yl))
  (cond ((> (- mxx mnx)(- mxy mny))
         (setq scale (/ 450 (- mxx mnx))))
        (t (setq scale (/ 450 (- mxy mny))))))
```

```
(defmethod (plot graph-pane)(xl yl)
  (do* ((xw xl (cdr xw))
        (yw yl (cdr yw))
        (x (car xw)(car xw))
        (y (car yw)(car yw)))
       ((null x))
    (graphics::draw-circle (+ 25 (round (* (- x mnx) scale)))
                           (- 475 (round (* (- y mny) scale)))
                           4
                           :stream self)))

(defmethod (draw-fit graph-pane)(xl yl)
  (let* ((a-b (CDR (.regres xl yl)))
         (a (car a-b))
         (b (cadr a-b))
         (pt1-y (+ a (* b mnx)))
         (pt2-y (+ a (* b mxx))))
    (graphics::draw-line (+ 25)
                         (- 475 (round (* (- pt1-y mny) scale)))
                         (+ 25 (round (* (- mxx mnx) scale)))
                         (- 475 (round (* (- pt2-y mny) scale)))
                         :thickness 2
                         :stream self)))

(defun test-fit (x y)
  (send *graph-pane* :expose)
  (send *graph-pane* :clear-window)
  (scale *graph-pane* x y)
  (plot *graph-pane* x y)
  (draw-fit *graph-pane* x y))

;;;     ***************************************************************************************************

;;;     ************************     ALO-H2O Admittance Calibration Equation Verifier     ****************

;;;     ***************************************************************************************************

;;;   The pressure and flow rate data below were manually recorded by Steve Morgan in July 1992 for use
;;;   in testing candidate admittance equations during the automatic admittance calibrator design phase.

(defun .test-press ()       ;;;    empty the tank !!!
  (SETQ *.PX-101* '(1.76 3.04 4.29)
        *.ST-ULLAGE* (format2 (/ (- *.ST-cap* 175) *.gal-to-cuft*))                ; =9.5 cuft
        *.st-up-admit* (.tankadmit *.ST-ULLAGE* *.PX-101* 30)                ; =2.27
        *.PX-109* '(1.06 2.29 3.51)
        *.VT-ULLAGE* (format2 (/ (- *.VT-cap* 10) *.gal-to-cuft*))                  ; =12 cuft
        *.vt-up-admit* (.tankadmit *.VT-ULLAGE* *.PX-109* 70)))              ; =1.78

(defun .test-depress ()      ;;;   Empty the tank!!!
  (SETQ *.ST-ULLAGE* (format2 (/ (- *.ST-cap* 175) *.gal-to-cuft*))                ; =9.5 cuft
        *.PX-101* '(12.2 11.3 10.4)
        *.st-vent-admit* (.tankadmit *.ST-ULLAGE* *.PX-101* 0)               ; =2.51
        *.VT-ULLAGE* (format2 (/ (- *.VT-cap* 10) *.gal-to-cuft*))                  ; =12 cuft
        *.PX-109* '(15.1 13.4 11.9)
        *.vt-vent-admit* (.tankadmit *.VT-ULLAGE* *.PX-109* 0)))             ; =5.09

(defun .test-Fast-Flow ()
  (SETQ *.fm-103*  '(10.3 13. 15.6)
        *.fm-102*  '(10.3 13. 15.6)
        *.px-101*  '(0. 0. 0.)
        *.dpx-104* '(2.39 2.22 2.07)
        *.px-115*  '(1. .2 0.)
        *.px-105*  '(23.4 32.8 44.)
        *.px-106*  '(20.6 28. 36.8)
        *.px-111*  '(5.7 9.1 13.1)
        *.px-107*  '(2.5 4.2 6.2)
        *.dpx-113* '(.49 .76 1.1)
        *.px-109*  '(0. .1 .2)
        *.suction-line-admit* (CAR (.stat (.admittances                              ;  =10.1
              *.fm-103* (.ad (/ -5 *.in-to-psi*) (mapcar #'- (mapcar #'+ *.px-101* *.dpx-104*) *.px-115*
)))))
        *.pump-to-4-way-admit* (CAR (.stat (.admittances
              *.fm-103* (.ad (/ 19 *.in-to-psi*)   (mapcar #'- *.px-105* *.px-115*)))))  ;  =2.2
        *.PX105-to-PX106-admit* (CAR (.stat (.admittances
              *.fm-103* (.ad (/ 24 *.in-to-psi*)   (mapcar #'- *.px-105* *.px-106*)))))  ;  =5.4
        *.transfer-line-admit* (CAR (.stat (.admittances
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
            *.fm-102* (.ad (/ -236 *.in-to-psi*) (mapcar #'- *.px-106* *.px-111*)))))  ;  =4.0
        *.ff-107-admit* (CAR (.stat (.admittances
            *.fm-102* (.ad (/ -1 *.in-to-psi*)   (mapcar #'- *.px-111* *.px-107*)))))  ;  =5.9
        *.final-fill-circuit-admit* (CAR (.stat (.admittances                          ;  =7.3
            *.fm-102* (.ad (/ -3 *.in-to-psi*) (mapcar #'- (mapcar #'- *.px-107* *.dpx-113*) *.px-109*
)))))))

(defun .test-ff108 ()
  (SETQ *.FM-102* '(10.3 13. 15.6)                                                  ;  =6.3
        *.px-111* '(6. 9.1 12.8)
        *.px-107* '(3.3 4.8 6.5)
        *.ff-108-admit* (CAR (.stat (.admittances
                                    *.FM-102* (.ad (/ -1 *.in-to-psi*) (mapcar #'- *.px-111* *.px-107*)
))))))

(defun .test-mcv101 ()
  (SETQ *.%open* '(.30 .40 .50 .60 .70 .80 .90 1.00)   ; = 2.7 * %open - 0.65 & r = 99.98%
        *.FM-101* '(.9 2.6 4.3 5.9 7.3 8.5 9.6 10.6)
        *.px-111* '(42.5 41.8 40.4 38.8 36.6 34.6 32.5 30.6)
        *.px-107* '(.6 .5 .8 1.1 1.4 2. 2.2 2.5)
        *.replenish-circuit-admit* (.regres *.%open* (.admittances
            *.FM-101* (.ad (/ -1 *.in-to-psi*) (mapcar #'- *.px-111* *.px-107*))))))

(defun .test-nozzle ()
  (SETQ *.px-107*  '(3.4 4.8 6.5)                                   ; =-1.
        *.FM-102*  '(9.6 12.7 15.6)
        *.px-109*  '(0. 0. .1)
        *.dpx-113* '(1.54 1.4 1.2)
        *.nozzle-admit* (CAR (.stat (.admittances (mapcar #'- *.FM-102* (.mul *.final-fill-circuit-admit*

          (mapcar #'sqrt (.ad (/ -3 *.in-to-psi*) (mapcar #'- *.px-107*
                                                  (mapcar #'+ *.px-109* *.dpx-113*)))))) ; vt-inl
et-flow
                                                (mapcar #'+ *.px-109* *.dpx-113*)))))) ; vt-inlet-pres

(defun .test-bleed ()
  (SETQ *.px-107*  '(3.5 5.1 7.1)                                ;  =-0.75
        *.FM-102*  '(9.6 12.7 15.5)
        *.px-109*  '(0. 0. .1)
        *.dpx-113* '(1.71 1.83 1.92)
        *.bleed-admit* (CAR (.stat (.admittances (mapcar #'- *.FM-102* (.mul *.final-fill-circuit-admit*
          (mapcar #'sqrt (.ad (/ -3 *.in-to-psi*) (mapcar #'- *.px-107*
                                                  (mapcar #'+ *.px-109* *.dpx-113*)))))) ; vt-inl
et-flow
                                               (mapcar #'+ *.px-109* *.dpx-113*)))))) ; vt-inlet-pres

(defun .test-recirc ()
  (SETQ *.%open*          '(.0   1.00)
        *.px-106sloflo*   '(23.8 32.1 43.2)     ;  SV-103 closed
        *.FM-103sloflo*   '(7.6  9.1  10.6)
        *.px-101*         '(0.   0.   0.)
        *.dpx-104*        '(2.04 2.04 2.04)
        *.px-106fasflo*   '(9.2  12.6 16.6)     ;  SV-103 open
        *.FM-103fasflo*   '(19.5 23.4 27.3)
        *.recirculation-admit* (.regres *.%open*    ;;;   admit = 5.6 * %open + 1.6 (r=99%)
         (LIST (CAR (.stat (.admittances
               *.FM-103sloflo* (mapcar #'- (mapcar #'- *.px-106sloflo* *.px-101*) *.dpx-104*))))
               (CAR (.stat (.admittances
               *.FM-103fasflo* (mapcar #'- (mapcar #'- *.px-106fasflo* *.px-101*) *.dpx-104*)))))))))

;;;     ************************************************************************************************

;;;     ************************     ALO-H2O Admittance Calibration Test Procedures     ******************

;;;     ************************************************************************************************

;;;   1.  Measure admittances of tank pressurization and depressurization lines with both tanks EMPTY.

(defprocedure .pressurize
  :pretty-name "Calibrating st- & vt-up-admit...."
  :actions
  ((L (format2 '(Calibrating st- & vt-up-admit....)))
   (L (SETQ *.ST-ULLAGE* (/ (- *.ST-cap* (GET-MODEL-VALUE 'ST-101 'level-out)) *.gal-to-cuft*)
            *.VT-ULLAGE* (/ (- *.VT-cap* (GET-MODEL-VALUE 'VT-101 'level-out)) *.gal-to-cuft*)))
;;;   Closed final-fill line (SV-111) prevents water loss.
   (C SV-101 OPEN    "Open storage tank's 30psi GN2 supply valve.")
```

```
    (C SV-114 OPEN    "Open vehicle tank's 70psi GN2 supply valve.")
    (WT 4)
    (L (SETQ *.a1* (GET-CURRENT-VALUE 'PX-101-M)
             *.b1* (GET-CURRENT-VALUE 'PX-109-M)))
    (WT 4)
    (L (SETQ *.a2* (GET-CURRENT-VALUE 'PX-101-M)
             *.b2* (GET-CURRENT-VALUE 'PX-109-M)))
    (WT 4)
    (L (setf *.a3* (GET-CURRENT-VALUE 'PX-101-M)
             *.b3* (GET-CURRENT-VALUE 'PX-109-M)))
    (L (SETQ *.st-up-admit* (.tankadmit *.ST-ULLAGE* (LIST *.a1* *.a2* *.a3*) *.ST-supl*)

             *.vt-up-admit* (.tankadmit *.VT-ULLAGE* (LIST *.b1* *.b2* *.b3*) *.VT-supl*)
             *.dummy* 1))  ;;;  Humor the balky "L" function
    (C SV-101 CLOSED  "Reinitialize.")
    (C SV-114 CLOSED  "Reinitialize.")))

(defprocedure .depressurize                    ;;;   Warning:  PX-101,2 not accurate above 25 psig !
  :pretty-name "Calibrating st- & vt-vent-admit...."
  :actions
  ((L (format2 '(Calibrating st- & vt-vent-admit....)))
   (L (SETQ *.ST-ULLAGE* (/ (- *.ST-cap* (GET-MODEL-VALUE 'ST-101 'level-out)) *.gal-to-cuft*)
            *.VT-ULLAGE* (/ (- *.VT-cap* (GET-MODEL-VALUE 'VT-101 'level-out)) *.gal-to-cuft*)))
;;;   Closed final-fill line (SV-111) prevents water loss.
   (C SV-106 OPEN    "Open storage tank vent.")
   (C SV-113 OPEN    "Open vehicle tank vent.")
   (WT 4)
   (L (SETQ *.a1* (GET-CURRENT-VALUE 'PX-101-M)
            *.b1* (GET-CURRENT-VALUE 'PX-109-M)))
   (WT 4)
   (L (SETQ *.a2* (GET-CURRENT-VALUE 'PX-101-M)
            *.b2* (GET-CURRENT-VALUE 'PX-109-M)))
   (WT 4)
   (L (setf *.a3* (GET-CURRENT-VALUE 'PX-101-M)
            *.b3* (GET-CURRENT-VALUE 'PX-109-M)))
;   (REPEAT 30 (WT 4)   ;;;   sample tank pressures every 4 seconds for 2 minutes
;          (L (setf *.PX-101* (CONS (GET-CURRENT-VALUE 'PX-101-M) *.PX-101*)
;                   *.PX-109* (CONS (GET-CURRENT-VALUE 'PX-109-M) *.PX-109*)))) ;;; "L" won't do this yet
   (L (SETQ *.st-vent-admit* (.tankadmit *.ST-ULLAGE* (LIST *.a1* *.a2* *.a3*) *.atm*)

            *.vt-vent-admit* (.tankadmit *.VT-ULLAGE* (LIST *.b1* *.b2* *.b3*) *.atm*)
            *.dummy* 1))
   (C SV-101 CLOSED  "Reinitialize.")
   (C SV-114 CLOSED  "Reinitialize.")))

;;;   Return the mean of all admittances, Ai, and print both their mean and standard deviation.
;;;   Bulk of gas in tank, Ni = N0 * Pi / P0, where N0 and P0 are initial ullage in cuft and absolute
;;;   pressure in psi.  Gas flow rate, Qi = (Ni - Ni-1) / Ts, where Ts is interval between pressure
;;;   samples in minutes.  Admittance, Ai = Qi / SQRT(Pi - Ps), where Ps is gas source pressure while
;;;   pressurizing or atm while depressurizing.  In other words,...
;;;                N0       Pi - Pi-1
;;;        Ai = ------ , -------------- , and P0 and Ts usually cancel out.
;;;             P0 Ts    SQRT(Pi - Ps)
(defun .tankadmit (ullage p Psource)
  (CAR (.stat (.admittances (.mul ullage (mapcar #'abs (.del (format2 p))))  ;  flows
                            (mapcar #'abs (.drop p Psource))))))             ;  dP

;;;   Replace any zero pressures with 1 micro-psi to humor the divide function.
(defun .humor (p)
  (mapcar #'(lambda (x) (max x 1.e-6)) p))

;;;   Subtract source pressure from every pressure sample.
(defun .drop (p Psource)
  (mapcar #'(lambda (x) (- x Psource)) p))

;;;   Find differences in consecutive pressure sample pairs.
(defun .del (p)
  (mapcar #'- p (cdr p)))

;;;   Get list of admittances = flows / SQRT (pressure drops).
(defun .admittances (flows pressures)
  (mapcar #'/ (format2 flows) (.humor (mapcar #'sqrt (format2 pressures)))))

;;;   2.  Calibrate Admittances Along the Fast Flow Path....

(defprocedure .FastFlow
```

```
    :pretty-name "Calibrating 6 fast-fill path admits...."
    :actions
    ((L (format2 '(Calibrating 6 fast-fill path admits....)))
     (C SV-106  OPEN       "Open storage tank vent.")
     (C SV-113  OPEN       "Open vehicle tank vent.")
     (C SV-104  OPEN       "Open pump circuit.")
     (C SV-107  OPEN       "Open main skid valve.")
     (C SV-111  OPEN       "Open final fill circuit.")
     (C RPM-101 (2500 rpm) "Pump slowly.")
     (WT 12)
     (L (SETQ *.a1* (GET-CURRENT-VALUE 'DPX-102-M) *.b1* (GET-CURRENT-VALUE 'PX-101-M)
              *.c1* (GET-CURRENT-VALUE 'DPX-104-M) *.d1* (GET-CURRENT-VALUE 'PX-115-M)
              *.e1* (GET-CURRENT-VALUE 'PX-105-M)  *.f1* (GET-CURRENT-VALUE 'PX-106-M)
              *.g1* (GET-CURRENT-VALUE 'PX-111-M)  *.h1* (GET-CURRENT-VALUE 'PX-107-M)
              *.i1* (GET-CURRENT-VALUE 'DPX-113-M) *.j1* (GET-CURRENT-VALUE 'PX-109-M)
              *.k1* (GET-CURRENT-VALUE 'DPX-103-M)))
     (C RPM-101 (3000 rpm) "Pump moderately.")
     (WT 4)
     (L (SETQ *.a2* (GET-CURRENT-VALUE 'DPX-102-M) *.b2* (GET-CURRENT-VALUE 'PX-101-M)
              *.c2* (GET-CURRENT-VALUE 'DPX-104-M) *.d2* (GET-CURRENT-VALUE 'PX-115-M)
              *.e2* (GET-CURRENT-VALUE 'PX-105-M)  *.f2* (GET-CURRENT-VALUE 'PX-106-M)
              *.g2* (GET-CURRENT-VALUE 'PX-111-M)  *.h2* (GET-CURRENT-VALUE 'PX-107-M)
              *.i2* (GET-CURRENT-VALUE 'DPX-113-M) *.j2* (GET-CURRENT-VALUE 'PX-109-M)
              *.k2* (GET-CURRENT-VALUE 'DPX-103-M)))
     (C RPM-101 (3500 rpm) "Pump rapidly.")
     (WT 4)
     (L (SETQ *.a3* (GET-CURRENT-VALUE 'DPX-102-M) *.b3* (GET-CURRENT-VALUE 'PX-101-M)
              *.c3* (GET-CURRENT-VALUE 'DPX-104-M) *.d3* (GET-CURRENT-VALUE 'PX-115-M)
              *.e3* (GET-CURRENT-VALUE 'PX-105-M)  *.f3* (GET-CURRENT-VALUE 'PX-106-M)
              *.g3* (GET-CURRENT-VALUE 'PX-111-M)  *.h3* (GET-CURRENT-VALUE 'PX-107-M)
              *.i3* (GET-CURRENT-VALUE 'DPX-113-M) *.j3* (GET-CURRENT-VALUE 'PX-109-M)
              *.k3* (GET-CURRENT-VALUE 'DPX-103-M)))
;;;   suction-line-admit's pressure drop = PX101 + DPX104 - PX115 - 5"
    (L (SETQ *.suction-line-admit*  (CAR (.stat (.admittances
                    (.mul *.FM103admit* (mapcar #'sqrt (LIST *.k1* *.k2* *.k3*)))     ; FM-103 flows
                    (.ad (/ -5 *.in-to-psi*) (mapcar #'- (mapcar #'+ (LIST *.b1* *.b2* *.b3*) (LIST *.c1*
*.c2* *.c3*))
                                             (LIST *.d1* *.d2* *.d3*))))))   ; PX101 + DPX104 - 5"
             *.pump-to-4-way-admit* (CAR (.stat (.admittances
                    (.mul *.FM103admit* (mapcar #'sqrt (LIST *.k1* *.k2* *.k3*)))     ; FM-103 flows
                    (.ad (/ 19 *.in-to-psi*) (mapcar #'- (LIST *.e1* *.e2* *.e3*)
                                             (LIST *.d1* *.d2* *.d3*))))))   ; PX105 - PX115 + 19"
             *.PX105-to-PX106-admit* (CAR (.stat (.admittances
                    (.mul *.FM103admit* (mapcar #'sqrt (LIST *.k1* *.k2* *.k3*)))     ; FM-103 flows
                    (.ad (/ 24 *.in-to-psi*) (mapcar #'- (LIST *.e1* *.e2* *.e3*)
                                             (LIST *.f1* *.f2* *.f3*))))))   ; PX105 - PX106 + 24"
             *.transfer-line-admit* (CAR (.stat (.admittances
                    (.mul *.FM102admit* (mapcar #'sqrt (LIST *.a1* *.a2* *.a3*)))     ; FM-102 flows
                    (.ad (/ -236 *.in-to-psi*) (mapcar #'- (LIST *.f1* *.f2* *.f3*)
                                               (LIST *.g1* *.g2* *.g3*))))))) ; PX106 - PX111 - 236"
             *.ff-107-admit* (CAR (.stat (.admittances
                    (.mul *.FM102admit* (mapcar #'sqrt (LIST *.a1* *.a2* *.a3*)))     ; FM-102 flows
                    (.ad (/ -1 *.in-to-psi*) (mapcar #'- (LIST *.g1* *.g2* *.g3*)
                                             (LIST *.h1* *.h2* *.h3*))))))   ; PX111 - PX107 - 1"
             *.final-fill-circuit-admit* (CAR (.stat (.admittances
                    (.mul *.FM102admit* (mapcar #'sqrt (LIST *.a1* *.a2* *.a3*)))     ; FM-102 flows
                    (.ad (/ -3 *.in-to-psi*) (mapcar #'- (mapcar #'- (LIST *.h1* *.h2* *.h3*) (LIST *.i1*
*.i2* *.i3*))
                                             (LIST *.j1* *.j2* *.j3*))))))   ; PX107 - DPX113 - PX109 - 3"
             *.dummy* 1))  ;  fake out the L-function
     (C RPM-101 (0 rpm) "Stop pumping.")
     (C SV-106  CLOSED  "Reinitialize.")
     (C SV-113  CLOSED  "Reinitialize.")
     (C SV-104  CLOSED  "Reinitialize.")
     (C SV-107  CLOSED  "Reinitialize.")
     (C SV-111  CLOSED  "Reinitialize.")))

;;;   3.  Redirect skid flow and calibrate the other two skid admittances.
(defprocedure .ff108
  :pretty-name "Calibrating SV-108's admittance...."
  :actions
  ((L (format2 '(Calibrating SV-108's admittance....)))
   (C SV-106  OPEN       "Open storage tank vent.")
   (C SV-113  OPEN       "Open vehicle tank vent.")
   (C SV-104  OPEN       "Open pump circuit.")
   (C SV-108  OPEN       "Open other skid valve.")
```

```
    (C SV-111  OPEN       "Open final fill circuit.")
    (C RPM-101 (3500 rpm) "Pump rapidly.")
    (WT 12)
    (L (SETQ *.a1* (GET-CURRENT-VALUE 'DPX-102-M)
             *.g1* (GET-CURRENT-VALUE 'PX-111-M)  *.h1* (GET-CURRENT-VALUE 'PX-107-M)))
    (C RPM-101 (3000 rpm) "Pump moderately.")
    (WT 4)
    (L (SETQ *.a2* (GET-CURRENT-VALUE 'DPX-102-M)
             *.g2* (GET-CURRENT-VALUE 'PX-111-M)  *.h2* (GET-CURRENT-VALUE 'PX-107-M)))
    (C RPM-101 (2500 rpm) "Pump slowly.")
    (WT 4)
    (L (SETQ *.a3* (GET-CURRENT-VALUE 'DPX-102-M)
             *.g3* (GET-CURRENT-VALUE 'PX-111-M)  *.h3* (GET-CURRENT-VALUE 'PX-107-M)))
    (L (SETQ *.ff-108-admit* (CAR (.stat (.admittances (.mul *.FM102admit* (mapcar #'sqrt (LIST *.a1* *.a2
* *.a3*))) ; FM-102 flow
          (.ad (/ -1 *.in-to-psi*) (mapcar #'- (LIST *.g1* *.g2* *.g3*) (LIST *.h1* *.h2* *.h3*)))))))))  ;
 PX111 - PX107 - 1"
    (C RPM-101 (0 rpm) "Stop pumping.")
    (C SV-106  CLOSED  "Reinitialize.")
    (C SV-113  CLOSED  "Reinitialize.")
    (C SV-104  CLOSED  "Reinitialize.")
    (C SV-108  CLOSED  "Reinitialize.")
    (C SV-111  CLOSED  "Reinitialize.")))

(defprocedure .mcv101
  :pretty-name "Calibrating MCV-101's admittance transfer function...."
  :actions
  ((L (format2 '(Calibrating MCV-101's admittance transfer function....)))
    (C SV-106  OPEN       "Open storage tank vent.")
    (C SV-113  OPEN       "Open vehicle tank vent.")
    (C SV-104  OPEN       "Open pump circuit.")
    (C SV-103  OPEN       "Speed up recirculation.")
    (C MCV-101 (.30 %)    "Open replenish valve to 30%")
    (C SV-111  OPEN       "Open final fill circuit.")
    (C RPM-101 (3500 rpm) "Pump rapidly.")
    (WT 25)
    (L (SETQ *.a1* (GET-CURRENT-VALUE 'DPX-101-M)
             *.g1* (GET-CURRENT-VALUE 'PX-111-M) *.h1* (GET-CURRENT-VALUE 'PX-107-M)))
    (C MCV-101 (.35 %)   "Open replenish valve to 35%")
    (WT 25)
    (L (SETQ *.a2* (GET-CURRENT-VALUE 'DPX-101-M)
             *.g2* (GET-CURRENT-VALUE 'PX-111-M) *.h2* (GET-CURRENT-VALUE 'PX-107-M)))
    (C MCV-101 (.40 %)   "Open replenish valve to 40%")
    (WT 25)
    (L (SETQ *.a3* (GET-CURRENT-VALUE 'DPX-101-M)
             *.g3* (GET-CURRENT-VALUE 'PX-111-M) *.h3* (GET-CURRENT-VALUE 'PX-107-M)))
    (L (setf admit (.admittances (.mul *.FM101admit* (mapcar #'sqrt (LIST *.a1* *.a2* *.a3*)))
                                 (.ad (/ -1 *.in-to-psi*) (mapcar #'- (LIST *.g1* *.g2* *.g3*) (LIST *.h1*
 *.h2* *.h3*))))
             *.replenish-circuit-admit* (.regres '(30. 35. 40.) admit)))
    (C RPM-101 (0. rpm) "Stop pumping.")
    (C SV-106  CLOSED  "Reinitialize.")
    (C SV-113  CLOSED  "Reinitialize.")
    (C SV-104  CLOSED  "Reinitialize.")
    (C SV-103  CLOSED  "Reinitialize.")
    (C SV-111  CLOSED  "Reinitialize.")
    (C MCV-101 (.0 %)  "Reinitialize.")))

;;;   4.  Calibrate nozzle and bleed valves.
(defprocedure .nozzle
  :pretty-name "Calibrating nozzle admittance...."
  :actions
  ((L (format2 '(Calibrating nozzle admittance....)))
    (C SV-106  OPEN       "Open storage tank vent.")
    (C SV-113  OPEN       "Open vehicle tank vent.")
    (C SV-104  OPEN       "Open pump circuit.")
    (C SV-107  OPEN       "Open main skid valve.")
    (C SV-111  OPEN       "Open final fill circuit.")
    (C SV-110  OPEN       "Open nozzle valve.")
    (C RPM-101 (3500 rpm) "Pump rapidly.")
    (WT 12)
    (L (SETQ *.a1* (GET-CURRENT-VALUE 'DPX-102-M) *.h1* (GET-CURRENT-VALUE 'PX-107-M)
             *.i1* (GET-CURRENT-VALUE 'DPX-113-M) *.j1* (GET-CURRENT-VALUE 'PX-109-M)))
    (C RPM-101 (3000 rpm) "Pump moderately.")
    (WT 4)
    (L (SETQ *.a2* (GET-CURRENT-VALUE 'DPX-102-M) *.h2* (GET-CURRENT-VALUE 'PX-107-M)
```

```
            *.i2* (GET-CURRENT-VALUE 'DPX-113-M) *.j2* (GET-CURRENT-VALUE 'PX-109-M)))
    (C RPM-101 (2500 rpm) "Pump slowly.")
    (WT 4)
    (L (SETQ *.a3* (GET-CURRENT-VALUE 'DPX-102-M) *.h3* (GET-CURRENT-VALUE 'PX-107-M)
            *.i3* (GET-CURRENT-VALUE 'DPX-113-M) *.j3* (GET-CURRENT-VALUE 'PX-109-M)))
;;;   Nozzle flow = FM102 - vt_inlet_flow, and
;;;   nozzle pressure drop = vt_inlet_pressure (i.e., the nozzle vents to the atmosphere).
    (L (SETQ *.nozzle-admit* (CAR (.stat (.admittances
      (mapcar #'- (.mul *.FM102admit* (mapcar #'sqrt (format2 (LIST *.a1* *.a2* *.a3*)))) (.vt-inlet-flow)
)   ;  flow
      (.vt-inlet-pressure))))))                                                    ;  dP
    (C RPM-101 (0 rpm) "Stop pumping.")
    (C SV-106  CLOSED  "Reinitialize SV106.")
    (C SV-113  CLOSED  "Reinitialize SV113.")
    (C SV-104  CLOSED  "Reinitialize SV104.")
    (C SV-107  CLOSED  "Reinitialize SV107.")
    (C SV-111  CLOSED  "Reinitialize SV111.")
    (C SV-110  CLOSED  "Reinitialize SV110.")))

;;;   Vt_inlet_pressure = PX109 + DPX113.
(defun .vt-inlet-pressure ()
  (mapcar #'+ (LIST *.j1* *.j2* *.j3*) (LIST *.i1* *.i2* *.i3*)))

;;;   Vt_inlet_flow = final_fill_circuit_admit * SQRT (PX107 - 3" - vt_inlet_pressure).
(defun .vt-inlet-flow ()
  (.mul *.final-fill-circuit-admit*
        (mapcar #'sqrt (.ad (/ -3 *.in-to-psi*) (mapcar #'- (LIST *.h1* *.h2* *.h3*) (.vt-inlet-pressure)
)))))

;;;   Multiply every element of a list by a scalar.
(defun .mul (multiplier x)
  (mapcar #'(lambda (x) (* x multiplier)) x))

;;;   Add a scalar to every element of a list.
(defun .ad (augend x)
  (mapcar #'(lambda (x) (+ x augend)) x))

(defprocedure .bleed
  :pretty-name "Calibrating bleed admittance...."
  :actions
  ((L (format2 '(Calibrating bleed admittance....)))
    (C SV-106  OPEN       "Open storage tank vent.")
    (C SV-113  OPEN       "Open vehicle tank vent.")
    (C SV-104  OPEN       "Open pump circuit.")
    (C SV-107  OPEN       "Open main skid valve.")
    (C SV-111  OPEN       "Open final fill circuit.")
    (C SV-109  OPEN       "Open bleed valve.")
    (C RPM-101 (3500 rpm) "Pump rapidly.")
    (WT 12)
    (L (SETQ *.a1* (GET-CURRENT-VALUE 'DPX-102-M) *.h1* (GET-CURRENT-VALUE 'PX-107-M)
            *.i1* (GET-CURRENT-VALUE 'DPX-113-M) *.j1* (GET-CURRENT-VALUE 'PX-109-M)))
    (C RPM-101 (3000 rpm) "Pump moderately.")
    (WT 4)
    (L (SETQ *.a2* (GET-CURRENT-VALUE 'DPX-102-M) *.h2* (GET-CURRENT-VALUE 'PX-107-M)
            *.i2* (GET-CURRENT-VALUE 'DPX-113-M) *.j2* (GET-CURRENT-VALUE 'PX-109-M)))
    (C RPM-101 (2500 rpm) "Pump slowly.")
    (WT 4)
    (L (SETQ *.a3* (GET-CURRENT-VALUE 'DPX-102-M) *.h3* (GET-CURRENT-VALUE 'PX-107-M)
            *.i3* (GET-CURRENT-VALUE 'DPX-113-M) *.j3* (GET-CURRENT-VALUE 'PX-109-M)))
;;;   Bleed flow = FM102 - vt_inlet_flow, and
;;;   bleed pressure drop = vt_inlet_pressure (i.e., the bleed valve vents to the atmosphere).
    (L (SETQ *.bleed-admit* (CAR (.stat (.admittances
      (mapcar #'- (.mul *.FM102admit* (mapcar #'sqrt (format2 (LIST *.a1* *.a2* *.a3*)))) (.vt-inlet-flow)
) ; flow
      (.vt-inlet-pressure))))))                                                    ; dP
    (C RPM-101 (0 rpm) "Stop pumping.")
    (C SV-106  CLOSED  "Reinitialize SV106.")
    (C SV-113  CLOSED  "Reinitialize SV113.")
    (C SV-104  CLOSED  "Reinitialize SV104.")
    (C SV-107  CLOSED  "Reinitialize SV107.")
    (C SV-111  CLOSED  "Reinitialize SV111.")
    (C SV-109  CLOSED  "Reinitialize SV109.")))

;;;   5.  Redirect pump flow, and calculate recirculation-admit.
(defprocedure .recirc
  :pretty-name "Calibrating recirculation path admittance...."
```

```
  :actions
  ((L (format2 '(Calibrating recirculation path admittance....)))
   (C SV-106  OPEN       "Open storage tank vent.")
   (C SV-113  OPEN       "Open vehicle tank vent.")
   (C SV-104  OPEN       "Open pump circuit.")
   (C SV-103  OPEN       "Speed up recirculation.")
   (C RPM-101 (3500 rpm) "Pump rapidly.")
   (WT 12)
   (L (SETQ *.a1* (GET-CURRENT-VALUE 'DPX-103-M) *.b1* (GET-CURRENT-VALUE 'PX-101-M)
            *.c1* (GET-CURRENT-VALUE 'DPX-104-M) *.f1* (GET-CURRENT-VALUE 'PX-106-M)))
   (C RPM-101 (3000 rpm) "Pump moderately.")
   (WT 4)
   (L (SETQ *.a2* (GET-CURRENT-VALUE 'DPX-103-M) *.b2* (GET-CURRENT-VALUE 'PX-101-M)
            *.c2* (GET-CURRENT-VALUE 'DPX-104-M) *.f2* (GET-CURRENT-VALUE 'PX-106-M)))
   (C RPM-101 (2500 rpm) "Pump slowly.")
   (WT 4)
   (L (SETQ *.a3* (GET-CURRENT-VALUE 'DPX-103-M) *.b3* (GET-CURRENT-VALUE 'PX-101-M)
            *.c3* (GET-CURRENT-VALUE 'DPX-104-M) *.f3* (GET-CURRENT-VALUE 'PX-106-M)))
   (L (SETQ *.fast-recirc* (CAR (.stat (.admittances
                              (.mul *.FM103admit* (mapcar #'sqrt (format2 (LIST *.a1* *.a2* *.a3*)))) ; F
M-103 flow
                              (mapcar #'- (mapcar #'- (LIST *.f1* *.f2* *.f3*) (LIST *.b1* *.b2* *.b3*))
                                      (LIST *.c1* *.c2* *.c3*)))))))            ; PX106 - PX101 - DPX104
   (C SV-103  CLOSED     "Slow down recirculation.")
   (L (SETQ *.a1* (GET-CURRENT-VALUE 'DPX-103-M) *.b1* (GET-CURRENT-VALUE 'PX-101-M)
            *.c1* (GET-CURRENT-VALUE 'DPX-104-M) *.f1* (GET-CURRENT-VALUE 'PX-106-M)))
   (C RPM-101 (3000 rpm) "Pump moderately.")
   (WT 4)
   (L (SETQ *.a2* (GET-CURRENT-VALUE 'DPX-103-M) *.b2* (GET-CURRENT-VALUE 'PX-101-M)
            *.c2* (GET-CURRENT-VALUE 'DPX-104-M) *.f2* (GET-CURRENT-VALUE 'PX-106-M)))
   (C RPM-101 (3500 rpm) "Pump rapidly.")
   (WT 4)
   (L (SETQ *.a3* (GET-CURRENT-VALUE 'DPX-103-M) *.b3* (GET-CURRENT-VALUE 'PX-101-M)
            *.c3* (GET-CURRENT-VALUE 'DPX-104-M) *.f3* (GET-CURRENT-VALUE 'PX-106-M)))
   (L (SETQ *.slow-recirc* (CAR (.stat (.admittances
                              (.mul *.FM103admit* (mapcar #'sqrt (format2 (LIST *.a1* *.a2* *.a3*)))) ; F
M-103 flow
                              (mapcar #'- (mapcar #'- (LIST *.f1* *.f2* *.f3*) (LIST *.b1* *.b2* *.b3*))
                                      (LIST *.c1* *.c2* *.c3*)))))))            ; PX106 - PX101 - DPX104
;;;   *.recirculation-admit* is a transfer function, mx+b, where m = 1 with SV103 open or 0 when closed:
   (L (SETQ *.recirculation-admit* (.regres '(0. 1.00)
                                            (LIST *.slow-recirc* *.fast-recirc*))))
   (C RPM-101 (0 rpm) "Stop pumping.")
   (C SV-106  CLOSED  "Reinitialize SV-106.")
   (C SV-113  CLOSED  "Reinitialize SV-113.")
   (C SV-104  CLOSED  "Reinitialize SV-104.")
   (C SV-103  CLOSED  "Reinitialize SV-103.")))

;;;    6.  Calibrate all derived admittances.

(defun .inseries (x y)
  (format2 (/ 1.0 (sqrt (+ (/ 1.0 (* x x)) (/ 1.0 (* y y)))))))

(defun .inparallel (x y)
  (format2 (+ x y)))

(defun .xferfcn (equation percent)              ;;;  equation of the form (y=__*x+__&r=__)
  (+ (* percent (CAR equation)) (CADR equation))) ;;;  percent ranges from 0. to 1.00

(defprocedure .derived
  :PRETTY-NAME "Calculating 5 derived admittances."
  :ACTIONS
  ((L (format2 '(Calibrating 5 derived admittances....)))
   (L (SETQ *.pump-circuit-admit* (.inseries *.pump-to-4-way-admit*
                                             (.inseries *.suction-line-admit* *.PX105-to-PX106-admit*))
            *.fast-fill-circuit-admit* *.ff-107-admit*  ;; only primary fast-fill valve is open usually
            *.skid-admit* (.inparallel *.fast-fill-circuit-admit* (.xferfcn *.replenish-circuit-admit* 0.3
3))
            *.upper-fill-circuit-admit* (.inseries *.skid-admit* *.final-fill-circuit-admit*)
            *.tank-fill-admit*          (.inseries *.upper-fill-circuit-admit* *.transfer-line-admit*)))))

;;; 7.  Print out all 14 admittances....

(defun .prtem ()
  (PROGn
    (format2 (LIST 'st-up-admit=              *.st-up-admit*))
```

```
    (format2 (LIST 'vt-up-admit=                *.vt-up-admit*))
    (format2 (LIST 'st-vent-admit=              *.st-vent-admit*))
    (format2 (LIST 'vt-vent-admit=              *.vt-vent-admit*))
    (format2 (LIST 'suction-line-admit=         *.suction-line-admit*))
    (format2 (LIST 'pump-to-4-way-admit=        *.pump-to-4-way-admit*))
    (format2 (LIST 'PX105-to-PX106-admit=       *.PX105-to-PX106-admit*))
    (format2 (LIST 'transfer-line-admit=        *.transfer-line-admit*))
    (format2 (LIST 'ff-107-admit=               *.ff-107-admit*))
    (format2 (LIST 'ff-108-admit=               *.ff-108-admit*))
    (format2 (LIST 'final-fill-circuit-admit=  *.final-fill-circuit-admit*))
    (format2 (LIST 'replenish-circuit-admit=   *.replenish-circuit-admit*))
    (format2 (LIST 'nozzle-admit=               *.nozzle-admit*))
    (format2 (LIST 'bleed-admit=                *.bleed-admit*))
    (format2 (LIST 'recirculation-admit=        *.recirculation-admit*))
    (format2 (LIST 'pump-circuit-admit=         *.pump-circuit-admit*))
    (format2 (LIST 'fast-fill-circuit-admit=   *.fast-fill-circuit-admit*))
    (format2 (LIST 'skid-admit=                 *.skid-admit*))
    (format2 (LIST 'upper-fill-circuit-admit= *.upper-fill-circuit-admit*))
    (format2 (LIST 'tank-fill-admit=            *.tank-fill-admit*))
    (format2 (LIST 'pump-xfr-fcn=               *.pump-xfer-fcn*))))

(defun .see-press ()
  (PROGn
    (format2 (LIST '*.a*= *.a1* *.a2* *.a3*))
    (format2 (LIST '*.b*= *.b1* *.b2* *.b3*))
    (format2 (LIST '*.c*= *.c1* *.c2* *.c3*))
    (format2 (LIST '*.d*= *.d1* *.d2* *.d3*))
    (format2 (LIST '*.e*= *.e1* *.e2* *.e3*))
    (format2 (LIST '*.f*= *.f1* *.f2* *.f3*))
    (format2 (LIST '*.g*= *.g1* *.g2* *.g3*))
    (format2 (LIST '*.h*= *.h1* *.h2* *.h3*))
    (format2 (LIST '*.i*= *.i1* *.i2* *.i3*))
    (format2 (LIST '*.j*= *.j1* *.j2* *.j3*))
    (format2 (LIST '*.k*= *.k1* *.k2* *.k3*))))

(defprocedure bob
              :actions
   ((L (setf aaaaa nil))
    (REPEAT 5
            (L (scl:beep))
            (WT 2)
            (L (setf aaaaa (cons 'ff aaaaa))))))

;;;  8.  Calibrate pump transfer function....

(defprocedure .pump-data
  :PRETTY-NAME "Calibrate pump transfer function."
  :ACTIONS
  ((L (format2 "Calibrating pump transfer function...."))
   (C SV-106  OPEN       "Open storage tank vent.")
   (C SV-113  OPEN       "Open vehicle tank vent.")
   (C SV-104  OPEN       "Open pump circuit.")
   (C SV-107  OPEN       "Open main skid valve.")
   (L (SETQ *.PX-105* nil *.pump-speed* 1800 *.PX-115* nil *.pmp-spd-list* nil))
   (C SV-111  OPEN       "Open final fill circuit.")
   (REPEAT 19
           (C RPM-101 *.pump-speed* "Ramp it up.")  ;; can't vary speed in a REPEAT yet.
           (WT 4)                                   ;; Steve Morgan, 15 July 1992
           (L (SETQ *.PX-105*       (CONS (GET-CURRENT-VALUE 'PX-105-M) *.PX-105*)
                    *.PX-115*       (CONS (GET-CURRENT-VALUE 'PX-115-M) *.PX-115*)
                    *.pmp-spd-list* (CONS *.pump-speed* *.pmp-spd-list*)
                    *.pump-speed*   (+ 100 *.pump-speed*)
                    *.dummy* 0)))
;  Add file access functions to save *.PX-115*, *.PX-115*, and *.pmp-spd-list* later.
   (C RPM-101 (0 rpm) "Stop pumping.")
   (C SV-106  CLOSED  "Close storage tank vent.")
   (C SV-113  CLOSED  "Close vehicle tank vent.")
   (C SV-104  CLOSED  "Shut down pump circuit.")
   (C SV-107  CLOSED  "Close main skid valve.")
   (C SV-111  CLOSED  "Close final fill circuit.")))

(defun .pump-analysis ()
;  Add file access functions to obtain *.PX-115*, *.PX-115*, and *.pmp-spd-list* later.
   (SETQ *.pump-xfer-fcn* (.regres *.pmp-spd-list* (.ad (/ 19 *.in-to-psi*) (mapcar #'- *.PX-105* *.PX-11
5*)))))
;   (test-fit (format2 *.pmp-spd-list*) (format2 (.ad (/ 19 *.in-to-psi*) (mapcar #'- *.PX-105* *.PX-115*
```

```
)))))))  ;; see it

(defprocedure .done
  :PRETTY-NAME "All done, thanks."
  :ACTIONS
  ((L (.prtem))))

(defun .test-it-all ()
  (PROGn
    (test-fit '(1 2.5 3) '(2 3 4))
    (.test-press)
    (.test-depress)
    (.test-Fast-Flow)
    (.test-ff108)
    (.test-mcv101)
    (.test-nozzle)
    (.test-bleed)
    (.test-recirc)))

(defprocedure .do-it-all
  :actions
    ((proc .pressurize)
;     (L    (.see-press))
     (proc .depressurize)
;     (L    (.see-press))
     (proc .fastflow)
;     (L    (.see-press))
     (proc .ff108)
;     (L    (.see-press))
     (proc .mcv101)
;     (L    (.see-press))
     (proc .nozzle)
;     (L    (.see-press))
     (proc .bleed)
;     (L    (.see-press))
     (proc .recirc)
;     (L    (.see-press))
     (proc .derived)
;     (L    (.see-press))
     (proc .done)))

;;;  ***************************************************************************************************

;;;  *************************  S T E V E ' S  S T A T I S T I C S  L I B R A R Y  ****************

;;;  ***************************************************************************************************

;;;  Optionally change to log (base e) or log-log graph...
(defun .logem (x)
  (mapcar 'log x))

;;;  Regression line: y = a + b x, r = ___...
(defun .regres (x y)
  (format2 (list (.slope (length x) (.sum x) (.sum y) (.sumsqr x) (.sumxy x y))
                 (.y-intcpt (length x) (.sum x) (.sum y) (.slope (length x) (.sum x) (.sum y) (.sumsqr x)
 (.sumxy x y)))
                 (.correl (length x) (.sum x) (.sum y) (.sumsqr x) (.sumsqr y) (.sumxy x y))
                 '(Y=__*X+__&R=__))))

;;;  Slope, b = (n * sumxy - sumx * sumy) / (n * sumxsqr - sumx * sumx)...
(defun .slope (n sx sy sx2 sxy)
  (/ (- (* n sxy) (* sx sy))
     (- (* n sx2) (* sx sx))))

;;;  Y-intercept, a = (sumy - b * sumx) / n...
(defun .y-intcpt (n sx sy b)
  (/ (- sy (* b sx))
     n))

;;;  Correlation, r = (n * sumxy - sumx * sumy) / sqrt[(n  * sumxsqr - sumx * sumx) ( n * sumysqr - sumy
* sumy)]...
(defun .correl (n sx sy sx2 sy2 sxy)
  (/ (- (* n sxy) (* sx sy))
     (sqrt (* (- (* n sx2) (* sx sx))
              (- (* n sy2) (* sy sy))))))
```

```
;;;  Mean * n...
(defun .sum (x)
  (cond ((null x) 0.0)
        ((atom x) x)
        (t (apply '+ (mapcar '.sum x)))))

;;;  Variance * n...
(defun .sumsqr (x)
  (cond ((null x) 0.0)
        ((atom x) (* x x))
        (t (apply '+ (mapcar '.sumsqr x)))))

;;;  Covariance...
(defun .sumxy (x y)
  (cond ((null x) 0.0)
        ((atom x) (* x y))
        (t (apply '+ (mapcar '.sumxy x y)))))

;;;   True Mean...
(defun .mean (x)
  (/ (.sum x) (length x)))

;;;   Standard Deviation...
(defun .sd (x)
  (- (/ (sqrt (.sumsqr x)) (length x)) (.mean x)))

;;;   Print elementary statistics....
(defun .stat (x)
  (format2 (list (.mean x)
                 (.sd   x)
                 '(mean=__ & s.d.=__))))

;;;  Reference:  JE Freund & RE Walpole: Mathematical Statistics, 3rd Ed., P-H, 1972.
```

APPENDIX B

CALIBRATOR OUTPUT LISTING

```
(CALIBRATING ST- & VT-UP-ADMIT....)
(CALIBRATING ST- & VT-UP-ADMIT....)
(CALIBRATING ST- & VT-UP-ADMIT....)
(CALIBRATING ST- & VT-UP-ADMIT....)
(1.3197069 2.6098394 3.8703947)
(12.209784 11.929865)
(28.680294 27.39016 26.129604)
(2.2796974 -0.6677079 (MEAN=__ & S.D.=__))
(1.2394838 2.4679117 3.6852846)
(15.271193 15.133762)
(68.76051 67.53209 66.31471)
(1.8416097 -0.539395 (MEAN=__ & S.D.=__))
(1.3197069 2.6098394 3.8703947)
(12.209784 11.929865)
(28.680294 27.39016 26.129604)
(2.2796974 -0.6677079 (MEAN=__ & S.D.=__))
(1.2394838 2.4679117 3.6852846)
(15.271193 15.133762)
(68.76051 67.53209 66.31471)
(1.8416097 -0.539395 (MEAN=__ & S.D.=__))
(CALIBRATING ST- & VT-VENT-ADMIT....)
(CALIBRATING ST- & VT-VENT-ADMIT....)
(CALIBRATING ST- & VT-VENT-ADMIT....)
(CALIBRATING ST- & VT-VENT-ADMIT....)
(3.7532854 3.2441072 2.771594)
(4.81885 4.4718537)
(3.7532854 3.2441072 2.771594)
(2.4850721 -0.7278601 (MEAN=__ & S.D.=__))
(3.6191034 2.8400812 2.153575)
(9.684409 8.534298)
(3.6191034 2.8400812 2.153575)
(5.0773726 -1.4871159 (MEAN=__ & S.D.=__))
(3.7532854 3.2441072 2.771594)
(4.81885 4.4718537)
(3.7532854 3.2441072 2.771594)
(2.4850721 -0.7278601 (MEAN=__ & S.D.=__))
(3.6191034 2.8400812 2.153575)
(9.684409 8.534298)
(3.6191034 2.8400812 2.153575)
(5.0773726 -1.4871159 (MEAN=__ & S.D.=__))
(CALIBRATING 6 FAST-FILL PATH ADMITS....)
(CALIBRATING 6 FAST-FILL PATH ADMITS....)
(CALIBRATING 6 FAST-FILL PATH ADMITS....)
(CALIBRATING 6 FAST-FILL PATH ADMITS....)
(13.699469 16.86502 19.976784)
(0.91344094 1.2903118 1.8511173)
(14.62123 -6.178755 (MEAN=__ & S.D.=__))
(13.699469 16.86502 19.976784)
(23.071518 33.07944 44.906635)
(2.9218197 -1.2346271 (MEAN=__ & S.D.=__))
(13.699469 16.86502 19.976784)
(5.7749076 8.752077 12.279726)
(5.7007403 -2.4094164 (MEAN=__ & S.D.=__))
(6.6288023 8.447623 10.216957)
(7.8893814 12.8127365 18.742004)
(2.3600085 -0.9974568 (MEAN=__ & S.D.=__))
(6.6288023 8.447623 10.216957)
(1.1949272 1.9406204 2.8330002)
(6.066092 -2.5638316 (MEAN=__ & S.D.=__))
(6.6288023 8.447623 10.216957)
(0.88412863 1.4795318 2.1698658)
(6.976916 -2.9486809 (MEAN=__ & S.D.=__))
(13.699469 16.86502 19.976784)
(0.91344094 1.2903118 1.8511173)
(14.62123 -6.178755 (MEAN=__ & S.D.=__))
(13.699469 16.86502 19.976784)
(23.071518 33.07944 44.906635)
(2.9218197 -1.2346271 (MEAN=__ & S.D.=__))
(13.699469 16.86502 19.976784)
(5.7749076 8.752077 12.279726)
(5.7007403 -2.4094164 (MEAN=__ & S.D.=__))
(6.6288023 8.447623 10.216957)
(7.8893814 12.8127365 18.742004)
(2.3600085 -0.9974568 (MEAN=__ & S.D.=__))
(6.6288023 8.447623 10.216957)
(1.1949272 1.9406204 2.8330002)
```

tank pressure
flow
dP

ditto
:

flow
dP

ditto
:

```
(6.066092 -2.5638316 (MEAN=__ & S.D.=__))
(6.6288023 8.447623 10.216957)
(0.88412863 1.4795318 2.1698658)
(6.976916 -2.9486809 (MEAN=__ & S.D.=__))
(CALIBRATING SV-108 'S ADMITTANCE....)
(CALIBRATING SV-108 'S ADMITTANCE....)
(CALIBRATING SV-108 'S ADMITTANCE....)
(CALIBRATING SV-108 'S ADMITTANCE....)
(10.22679 8.358611 6.4019327)
(2.5856705 1.735714 1.0150461)
(6.3529034 -2.685051 (MEAN=__ & S.D.=__))
(10.22679 8.358611 6.4019327)
(2.5856705 1.735714 1.0150461)
(6.3529034 -2.685051 (MEAN=__ & S.D.=__))
(CALIBRATING MCV-101 'S ADMITTANCE TRANSFER FUNCTION....)
(CALIBRATING MCV-101 'S ADMITTANCE TRANSFER FUNCTION....)
(CALIBRATING MCV-101 'S ADMITTANCE TRANSFER FUNCTION....)
(CALIBRATING MCV-101 'S ADMITTANCE TRANSFER FUNCTION....)
(0.37875423 0.7495712 1.1039773)
(7.905173 7.5464616 7.149731)
(0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__))
(0.37875423 0.7495712 1.1039773)
(7.905173 7.5464616 7.149731)
(0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__))
(0.37875423 0.7495712 1.1039773)
(7.905173 7.5464616 7.149731)
(0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__))
(0.37875423 0.7495712 1.1039773)
(7.905173 7.5464616 7.149731)
(0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__))
(CALIBRATING NOZZLE ADMITTANCE....)
(CALIBRATING NOZZLE ADMITTANCE....)
(CALIBRATING NOZZLE ADMITTANCE....)
(CALIBRATING NOZZLE ADMITTANCE....)
(7.8357015 5.238446 3.0496871)
(-0.0586195 0.065597534 0.042866707)
(0.5412985 0.5452448 0.52057886)
(0.02285788 0.021576738 (MEAN=__ & S.D.=__))
(7.8357015 5.238446 3.0496871)
(-0.0586195 0.065597534 0.042866707)
(0.5412985 0.5452448 0.52057886)
(0.02285788 0.021576738 (MEAN=__ & S.D.=__))
(CALIBRATING BLEED ADMITTANCE....)
(CALIBRATING BLEED ADMITTANCE....)
(CALIBRATING BLEED ADMITTANCE....)
(CALIBRATING BLEED ADMITTANCE....)
(7.8169947 5.2061214 3.0175722)
(0.055433273 0.071427345 0.036339283)
(0.62161875 0.6242012 0.59833926)
(0.0692315 -0.027968463 (MEAN=__ & S.D.=__))
(7.8169947 5.2061214 3.0175722)
(0.055433273 0.071427345 0.036339283)
(0.62161875 0.6242012 0.59833926)
(0.0692315 -0.027968463 (MEAN=__ & S.D.=__))
(CALIBRATING RECIRCULATION PATH ADMITTANCE....)
(CALIBRATING RECIRCULATION PATH ADMITTANCE....)
(CALIBRATING RECIRCULATION PATH ADMITTANCE....)
(CALIBRATING RECIRCULATION PATH ADMITTANCE....)
(30.0 30.0 28.974215)
(20.539597 20.539597 20.185389)
(15.253061 11.207141 7.769389)
(6.2120996 -2.5950906 (MEAN=__ & S.D.=__))
(30.0 30.0 28.974215)
(20.539597 20.539597 20.185389)
(15.253061 11.207141 7.769389)
(6.2120996 -2.5950906 (MEAN=__ & S.D.=__))
(4.361521 6.2761393 8.543359)
(7.8315954 9.394584 10.960884)
(21.331217 30.704245 41.794704)
(1.695516 -0.71660936 (MEAN=__ & S.D.=__))
(4.361521 6.2761393 8.543359)
(7.8315954 9.394584 10.960884)
(21.331217 30.704245 41.794704)
(1.695516 -0.71660936 (MEAN=__ & S.D.=__))
(4.5165834 1.6955161 1.0 (Y=__*X+__&R=__))
(4.5165834 1.6955161 1.0 (Y=__*X+__&R=__))
```

```
(4.5165834 1.6955161 1.0 (Y=__*X+__&R=__))
(4.5165834 1.6955161 1.0 (Y=__*X+__&R=__))
(CALIBRATING 5 DERIVED ADMITTANCES....)
(CALIBRATING 5 DERIVED ADMITTANCES....)
(CALIBRATING 5 DERIVED ADMITTANCES....)
(CALIBRATING 5 DERIVED ADMITTANCES....)
5.31131
2.5600228
5.375187
4.2580447
2.0641644
5.31131
2.5600228
5.375187
4.2580447
2.0641644
(ST-UP-ADMIT= 2.2796974)
(VT-UP-ADMIT= 1.8416097)
(ST-VENT-ADMIT= 2.4850721)
(VT-VENT-ADMIT= 5.0773726)
(SUCTION-LINE-ADMIT= 14.62123)
(PUMP-TO-4-WAY-ADMIT= 2.9218197)
(PX105-TO-PX106-ADMIT= 5.7007403)
(TRANSFER-LINE-ADMIT= 2.3600085)
(FF-107-ADMIT= 6.066092)
(FF-108-ADMIT= 6.3529034)
(FINAL-FILL-CIRCUIT-ADMIT= 6.976916)
(REPLENISH-CIRCUIT-ADMIT= (0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__)))
(NOZZLE-ADMIT= 0.02285788)
(BLEED-ADMIT= 0.0692315)
(RECIRCULATION-ADMIT= (4.5165834 1.6955161 1.0 (Y=__*X+__&R=__)))
(PUMP-CIRCUIT-ADMIT= 2.5600228)
(FAST-FILL-CIRCUIT-ADMIT= 6.066092)
(SKID-ADMIT= 5.375187)
(UPPER-FILL-CIRCUIT-ADMIT= 4.2580447)
(TANK-FILL-ADMIT= 2.0641644)
(PUMP-XFR-FCN= NIL)        not implemented
(ST-UP-ADMIT= 2.2796974)
(VT-UP-ADMIT= 1.8416097)
(ST-VENT-ADMIT= 2.4850721)
(VT-VENT-ADMIT= 5.0773726)
(SUCTION-LINE-ADMIT= 14.62123)
(PUMP-TO-4-WAY-ADMIT= 2.9218197)
(PX105-TO-PX106-ADMIT= 5.7007403)
(TRANSFER-LINE-ADMIT= 2.3600085)
(FF-107-ADMIT= 6.066092)
(FF-108-ADMIT= 6.3529034)
(FINAL-FILL-CIRCUIT-ADMIT= 6.976916)
(REPLENISH-CIRCUIT-ADMIT= (0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__)))
(NOZZLE-ADMIT= 0.02285788)
(BLEED-ADMIT= 0.0692315)
(RECIRCULATION-ADMIT= (4.5165834 1.6955161 1.0 (Y=__*X+__&R=__)))
(PUMP-CIRCUIT-ADMIT= 2.5600228)
(FAST-FILL-CIRCUIT-ADMIT= 6.066092)
(SKID-ADMIT= 5.375187)
(PUMP-XFR-FCN= NIL)
(ST-UP-ADMIT= 2.2796974)
(VT-UP-ADMIT= 1.8416097)
(ST-VENT-ADMIT= 2.4850721)
(VT-VENT-ADMIT= 5.0773726)
(SUCTION-LINE-ADMIT= 14.62123)
(PUMP-TO-4-WAY-ADMIT= 2.9218197)
(PX105-TO-PX106-ADMIT= 5.7007403)
(TRANSFER-LINE-ADMIT= 2.3600085)
(FF-107-ADMIT= 6.066092)
(FF-108-ADMIT= 6.3529034)
(FINAL-FILL-CIRCUIT-ADMIT= 6.976916)
(REPLENISH-CIRCUIT-ADMIT= (0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__)))
(NOZZLE-ADMIT= 0.02285788)
(BLEED-ADMIT= 0.0692315)
(RECIRCULATION-ADMIT= (4.5165834 1.6955161 1.0 (Y=__*X+__&R=__)))
(PUMP-CIRCUIT-ADMIT= 2.5600228)
(FAST-FILL-CIRCUIT-ADMIT= 6.066092)
(SKID-ADMIT= 5.375187)
```

```
(UPPER-FILL-CIRCUIT-ADMIT= 4.2580447)
(TANK-FILL-ADMIT= 2.0641644)
(PUMP-XFR-FCN= NIL)
(ST-UP-ADMIT= 2.2796974)
(VT-UP-ADMIT= 1.8416097)
(ST-VENT-ADMIT= 2.4850721)
(VT-VENT-ADMIT= 5.0773726)
(SUCTION-LINE-ADMIT= 14.62123)
(PUMP-TO-4-WAY-ADMIT= 2.9218197)
(PX105-TO-PX106-ADMIT= 5.7007403)
(TRANSFER-LINE-ADMIT= 2.3600085)
(FF-107-ADMIT= 6.066092)
(FF-108-ADMIT= 6.3529034)
(FINAL-FILL-CIRCUIT-ADMIT= 6.976916)
(REPLENISH-CIRCUIT-ADMIT= (0.027816162 -0.7000845 0.99999374 (Y=__*X+__&R=__)))
(NOZZLE-ADMIT= 0.02285788)
(BLEED-ADMIT= 0.0692315)
(RECIRCULATION-ADMIT= (4.5165834 1.6955161 1.0 (Y=__*X+__&R=__)))
(PUMP-CIRCUIT-ADMIT= 2.5600228)
(FAST-FILL-CIRCUIT-ADMIT= 6.066092)
(SKID-ADMIT= 5.375187)
(UPPER-FILL-CIRCUIT-ADMIT= 4.2580447)
(TANK-FILL-ADMIT= 2.0641644)
(PUMP-XFR-FCN= NIL)
```

REFERENCES

[1] Steve Morgan: *KATE's Model Verification Tools*. University of Central Florida's 1992 NASA/ASEE Summer Faculty Fellowship Program, Contractor Report No. NASA-NGT-60002, Supplement 6, August 9, 1992.

[2] John K. Vennard: *Elementary Fluid Mechanics*. 4th Ed., New York, Wiley, 1963, p.280.

[3] Bob Merchant: "The initial phase of the admittance calculator." In G:>KATE>DOC>ALO>ADMITTANCE-CALCULATOR.LISP.9, April 5, 1992.

[4] Bob Merchant's ALO Knowledge Base File G:>KATE>ALO-KB>FLUIDS-PRESSURES.LISP.18, March 4, 1992.

[5] Bob Merchant's ALO Knowledge Base File G:>KATE>ALO-KB>ADMITTANCES .LISP.23, June 4, 1992.

[6] Bob Merchant's ALO Knowledge Base File G:>KATE>ALO-KB>FLOW-RATES-PRESSURES.LISP.15, January 20, 1992.

[7] JE Freund & RE Walpole: "Mathematical Statistics," 3rd Ed., Prentice-Hall, 1972.

[8] Bob Merchant's ALO Control Procedures File: G:>KATE>ALO-KB>CONTROL-PROCEDURES .LISP.41, March 6, 1992.

[9] Steve Morgan's Automatic Admittance Calibrator Program Listing: G:>MORGAN>LAST.LISP.7, August 6, 1992.

[10] Steve Morgan's Calibrator Output Listing: G:>MORGAN>ANSWERS.LISP.6, August 6, 1992.

[11] Bob Merchant's ALO Knowledge Base File G:>KATE>ALO-KB>ALO-H2O.LISP.37, May 8, 1992.

[12] Bob Merchant: "ALO Admittance Plots And Calculations Notebook," 63pp., January 1, 1992.

[13] Steve Beltz' Personal Communication on "Sensor Tolerances." July 15, 1992.